Universidade de Aveiro
2015

Departamento de Eletrónica, Telecomunicações e Informática

BRUNO MIGUEL DA SILVA PEREIRA

SOLUÇÃO SAAS DE GESTÃO DE ARRENDAMENTOS DE IMÓVEIS

Universidade de Aveiro
2015

Departamento de Eletrónica, Telecomunicações e Informática

BRUNO MIGUEL DA SILVA PEREIRA

SOLUÇÃO SAAS DE GESTÃO DE ARRENDAMENTOS DE IMÓVEIS

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Sistemas de Informação, realizada sob a orientação do Dr. Cláudio Teixeira, Equiparado a Investigador Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

Dedico este trabalho aos meus pais.

o júri

presidente

Prof. Doutor José Luís Guimarães Oliveira

Professor Associado da Universidade de Aveiro

vogais

Prof. Doutor André Frederico Guilhoto Monteiro

Professor Auxiliar Convidado no Instituto Superior Miguel Torga

Doutor Cláudio Jorge Vieira Teixeira

Equiparado a Investigador Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos

Em primeiro lugar quero agradecer a minha família, aos meus pais, irmã, Joana, e todos que de certo modo, me ajudaram e permitiram chegar até este momento.

Ao Professor Cláudio Teixeira pelo apoio e disponibilidade ao longo destes últimos anos.

Aos meus amigos, aos colegas do IEETA e do Clube dos Galitos por todos os momentos que passámos juntos.

A todos, o meu muito obrigado.

palavras-chave Imóvel, serviço, sistemas de informação, software, faturação, arrendamentos.

resumo Esta dissertação tem como objetivo o desenvolvimento de um serviço de arrendamento de imóveis com faturação *online* integrada. Pretende-se que qualquer cliente, seja ele uma pessoa singular ou uma empresa, possa utilizar este serviço para gerir todas as operações que o arrendamento de um imóvel envolve, desde a publicitação de imóveis até à gestão de contratos de imóveis.

keywords property, service, information system, software, invoicing, rental.

abstract This dissertation aims to develop a property rental service with integrated online billing. It is intended that any customer, whether an individual or a company, can use this service to manage all the operations that the rental of a property involves, since the advertisement of the property till the management of the contracts.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ACRÓNIMOS

API	**A**pplication **P**rogramming **I**nterface
SaaS	**S**oftware **a**s **a** **S**ervice
MVC	**M**odel **V**iew **C**ontroller
MSSQL	**M**icrosoft **SQL** **S**erver
HTTP	**H**ypertext **T**ransfer **P**rotocol
PDF	**P**ortable **D**ocument **F**ormat
JSON	**J**avaScript **O**bject **N**otation
CSS	**C**ascading **S**tyle **S**heets
URL	**U**niform **R**esource **L**ocator
SSL	**S**ecure **S**ockets **L**ayer
AT	**A**utoridade **T**ributária e Aduaneira

1. INTRODUÇÃO

1.1 ENQUADRAMENTO

Cada vez mais vemos a substituição de sistemas de gestão tradicionais por soluções *online*. Estas trazem vantagens de nos permitirem desfazer de encargos de infraestruturas e oferecem novas metodologias de pagamento como o *pay-as-you-go*. Estas soluções baseiam-se num modelo de distribuição e comercialização de *software* denominada de *software as a service*. Neste modelo o cliente apenas consome o serviço que necessita e paga o valor do serviço que recebe, ficando toda a responsabilidade da disponibilização de serviço e infraestrutura do lado do fornecedor, tornando-se assim mais apelativa relativamente a outros tipo de sistemas mais tradicionais.

Um dos setores de negócio que apresenta uma falta de serviços de gestão *online* é o Imobiliário, mais especificamente a gestão de arrendamento. De acordo com uma das maiores páginas web de anúncios de imóveis, os apartamentos são o imóvel com maior procura por parte dos consumidores (Figura 1). Foi realizado um estudo sobre a finalidade da procura imobiliária (Figura 2), e após a análise do mesmo é possível afirmar que cerca de 64% das pessoas procuram apartamentos para arrendamento e não para compra. Posto isto, podemos afirmar que os arrendamentos são um dos principais modelos de negócios de imóveis em Portugal.

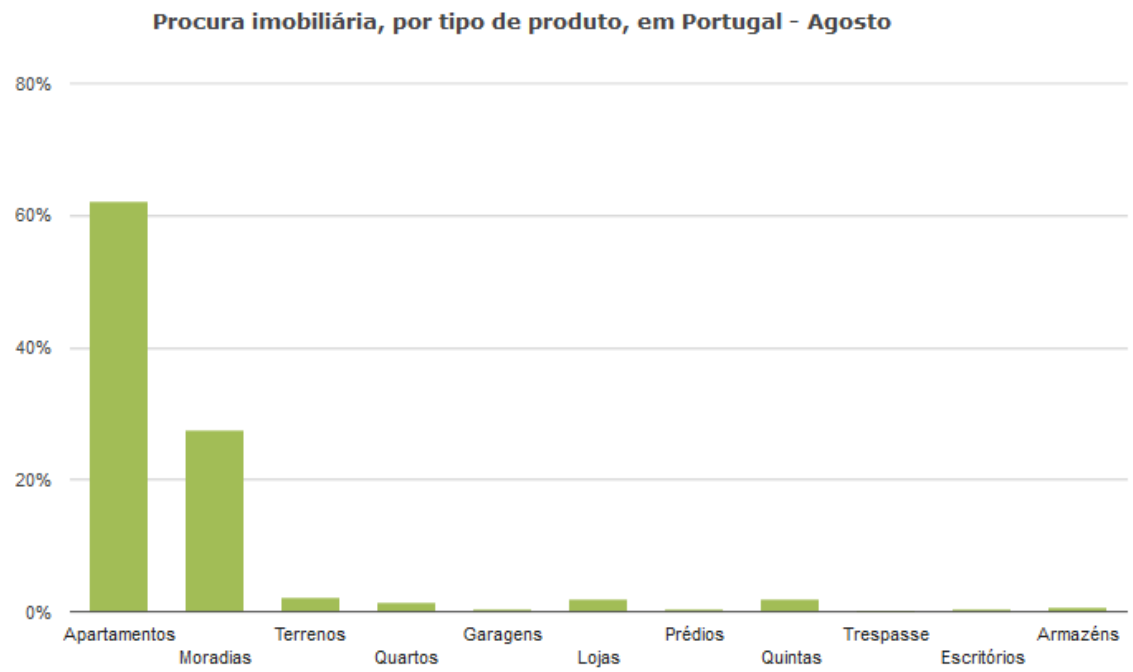


Figura 1 - Procura imobiliária, por tipo de produto, em Portugal – Agosto 2015 [1]

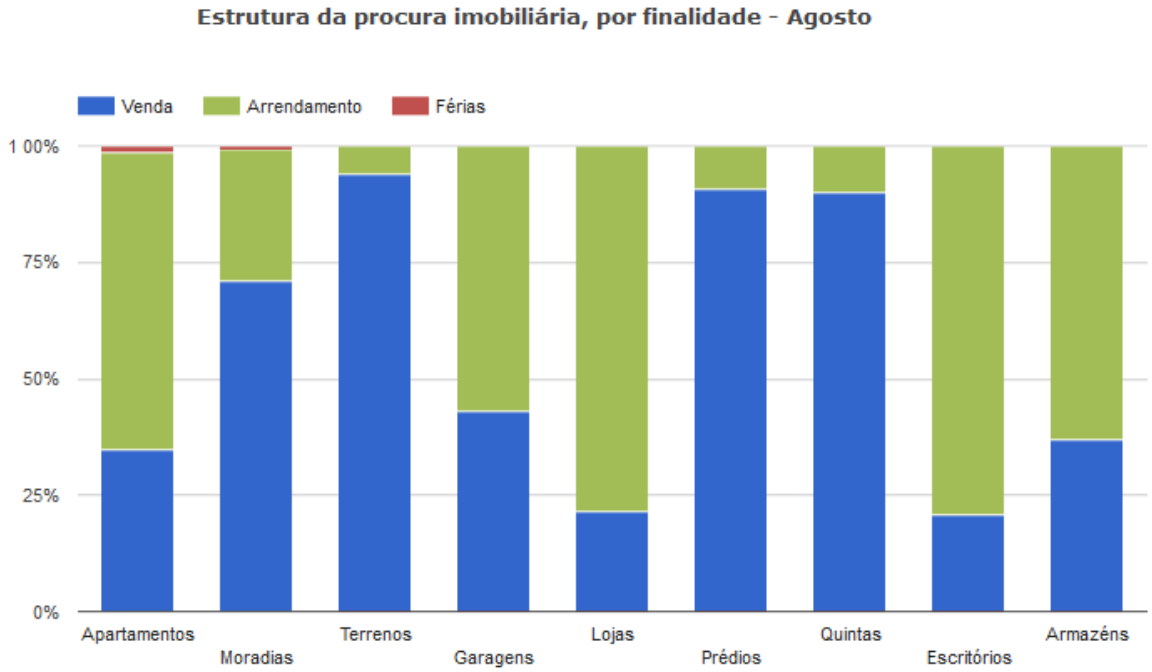


Figura 2 - Estrutura da procura imobiliária, por finalidade – Agosto [2]

1.2 MOTIVAÇÃO

Atualmente em Portugal, o setor imobiliário é composto por empresas mediadoras ou por independentes. Existem já algumas ofertas *online* que permitem a publicitação de imóveis. Estas ofertas, são sistemas privados desenvolvidos exclusivamente para suportar as necessidades de uma empresa. Além destas, existem ainda plataformas de publicitação abertas a qualquer utilizador como é o caso da Imovirtual. Para complementar essas plataformas, a nível de faturação e gestão de contratos, as empresas recorrem a outras soluções, mas na maioria dos casos tanto para independentes como para empresas, os custos e a complexidade desses meios podem ser demasiado elevados.

Avaliando esta realidade e de modo a simplificar o processo de gestão de arrendamento consideramos que existe a falta de uma solução que permita fazer a realização de forma mais simples de todas as etapas de um processo de arrendamento (desde a publicitação até à finalização do contrato), tudo na mesma plataforma.

1.3 OBJETIVOS

Tal como foi referido anteriormente já existem algumas aplicações *online* que permitem a um utilizador a publicitação online de imóveis, no entanto não existe nenhuma solução *online* que permita a essa entidade fazer todo o processo de gestão de arrendamento de um imóvel, incluindo a faturação sem a utilização de infraestruturas próprias.

Pretende-se então desenvolver um serviço *online* que consiga realizar todo esse processo. Uma solução que permita a um conjunto de entidades publicitar, gerir e realizar toda a faturação envolvida com um conjunto de imóveis. A solução irá permitir que utilizadores se registem na aplicação como clientes para que possam fazer pedidos de arrendamentos e gerir os seus contratos.

Este serviço deverá permitir que vários utilizadores se registem como entidades de arrendamento e que as suas regras e toda a gestão de imóveis seja independe a cada entidade.

1.4 CONTRIBUIÇÃO

Ao longo desta dissertação pretende-se desenvolver uma aplicação que fornece um serviço que facilite e torne possível a realização do processo de arrendamento de imóveis incluindo faturação, tudo numa única solução *online*. Uma única solução para ser utilizada tanto por inquilinos como por empresas de gestão de arrendamentos ou qualquer outro tipo de proprietários de imóveis, facilitando a comunicação entre elas. Numa primeira fase irão ser analisadas as soluções já existentes e as suas funcionalidades. Seguidamente iremos analisar os requisitos e as tecnologias necessárias para o desenvolvimento do serviço. Por fim será desenvolvido um protótipo que executa as operações principais necessárias para a realização do serviço proposto.

1.5 ORGANIZAÇÃO DA DISSERTAÇÃO

Esta dissertação está dividida em cinco capítulos, sendo este o primeiro. O segundo capítulo aborda o estado de arte, apresentando algumas soluções, existentes em Portugal, que se enquadram na área de desenvolvimento desta dissertação. No terceiro foram analisados os requisitos, apresentada a arquitetura bem como o modelo de dados e por fim, foi ainda apresentado o modo de implementação. Já no quarto foi demonstrado o protótipo desenvolvido e as suas funcionalidades. No quinto capítulo foi realizada uma análise ao trabalho feito e deixadas ideias relativamente a um possível trabalho futuro.

2. ESTADO DE ARTE

De modo a fazer um levantamento de necessidades para este serviço é necessário compreender e analisar as diversas soluções já existentes no setor imobiliário. Para isso, neste capítulo, iremos analisar algumas soluções relacionadas com arrendamentos e ainda algumas soluções de faturação a *online* através de API's.

2.1 SOLUÇÕES DE ARRENDAMENTO

Na área dos arrendamentos, existem soluções que têm algumas das funcionalidades que são pretendidas na solução que será desenvolvida. Neste subcapítulo vamos analisar algumas dessas soluções.

2.1.1 UniPlaces

UniPlaces é uma solução *online* para pesquisa e arrendamento de casas ou quartos a estudantes. Está solução permite aos estudantes encontrar um imóvel para arrendar durante estadias de médio prazo, normalmente a duração dos períodos escolares, assim como permite aos arrendatários publicitarem-nos [3] [4].

De modo a obter rendimento, o UniPlaces usa uma taxa de serviço que é paga quando um estudante faz o a reserva do quarto [5]. Para isso tem integrado um sistema de pagamentos, isto é, quando é feita uma reserva o pagamento do primeiro mês de renda tem que ser efetuado. Este pagamento é retido pelo UniPlaces e dois dias depois do estudante se acomodar na casa ou quarto o montante referente a renda é transferido para o dono do imóvel ficando a taxa de serviço para o Uniplaces (Figura 4). A partir desta fase, a gestão do imóvel e pagamentos é feita fora do UniPlaces. UniPlaces é uma das soluções que tem demonstrado maiores parecenças com a solução que pretendemos desenvolver,

sendo que depois da fase de pagamentos, na nossa solução, pretende-se continuar a dar suporte ao inquilino e à entidade que gere o imóvel.

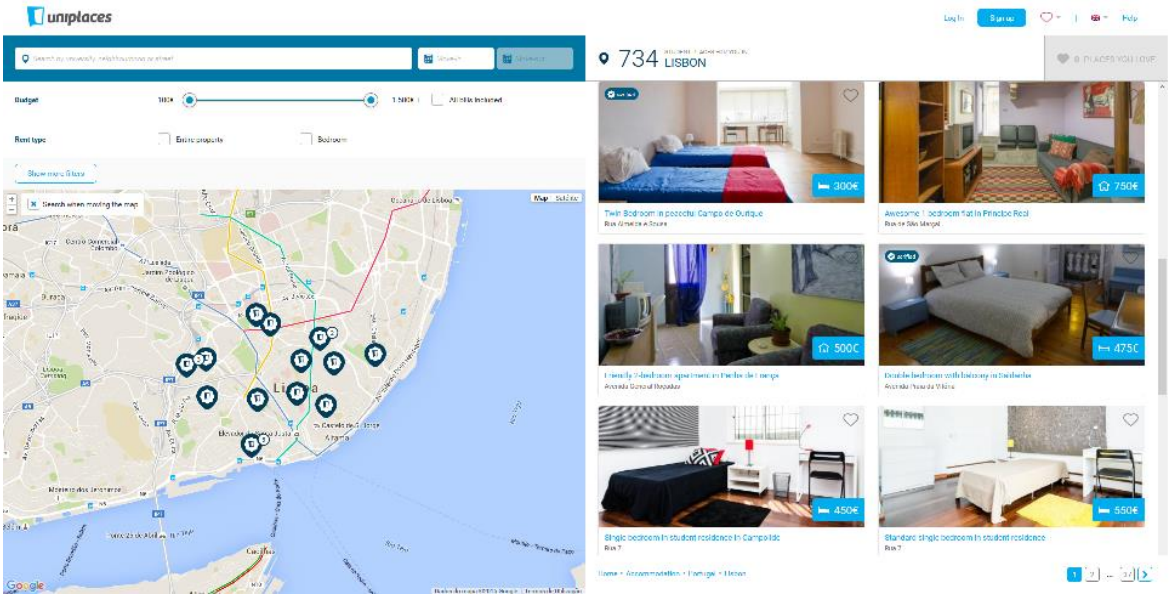


Figura 3 - UniPlaces, pesquisa de quartos [4]

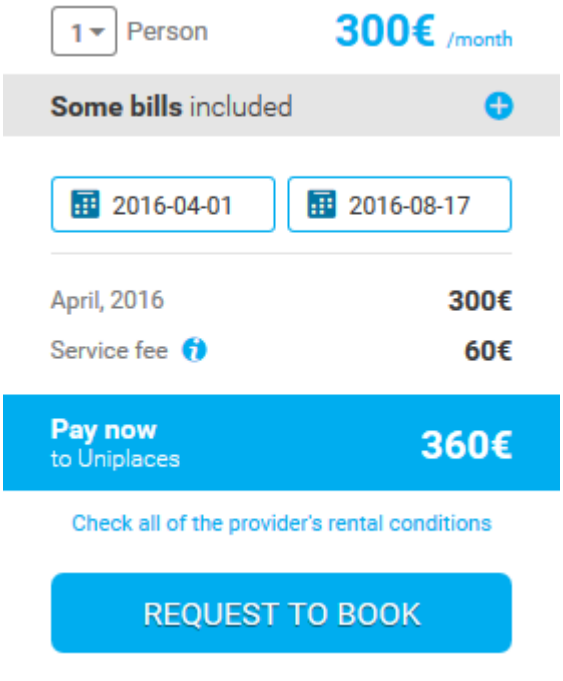


Figura 4 - UniPlaces, Service fees e Pagamento. [4]

2.1.2 Imovirtual

Imovirtual é uma solução *online* para publicação de anúncios de imóveis. O Imovirtual permite a um utilizador criar, seja ele independente ou uma empresa, um conjunto de anúncios de qualquer tipo de imóveis. O anúncio do imóvel, além da informação que o anunciante disponibilizar do imóvel, contém a informação necessária para que uma pessoa interessada no mesmo possa entrar em contacto com a entidade responsável por ele [6].

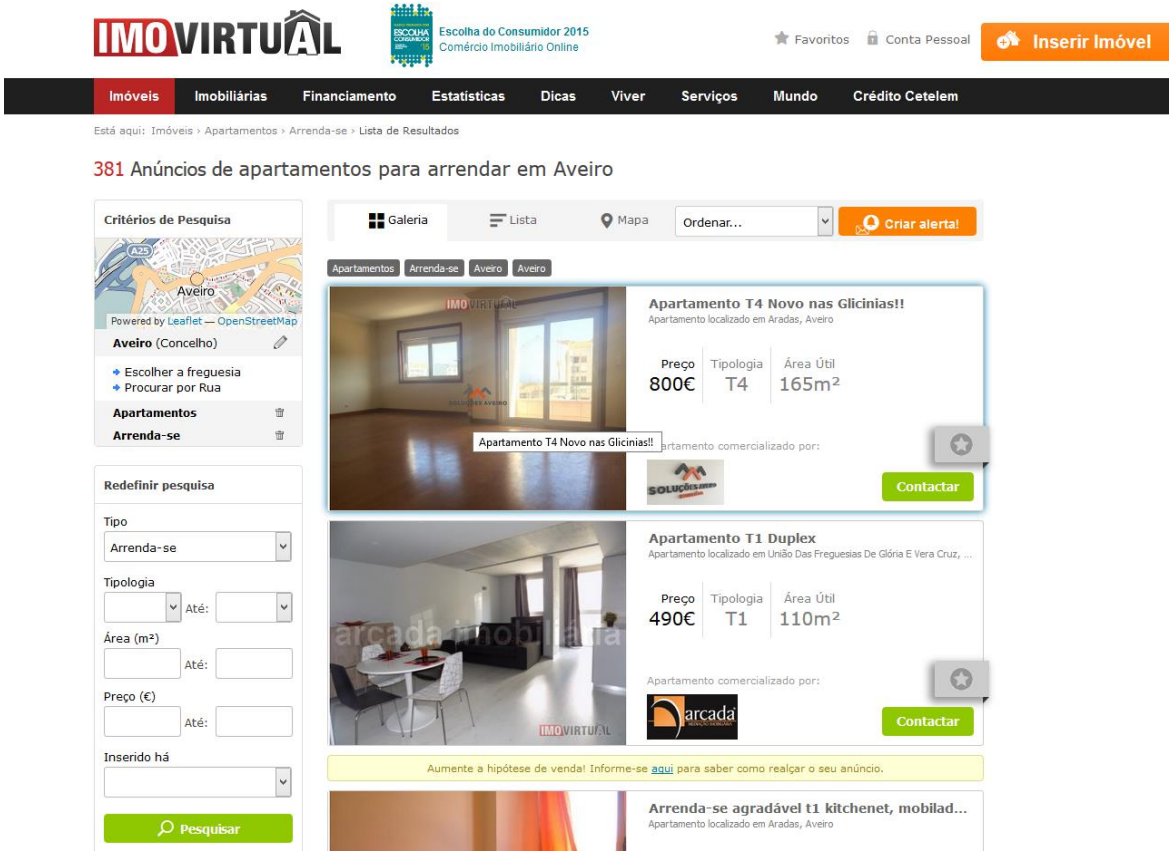
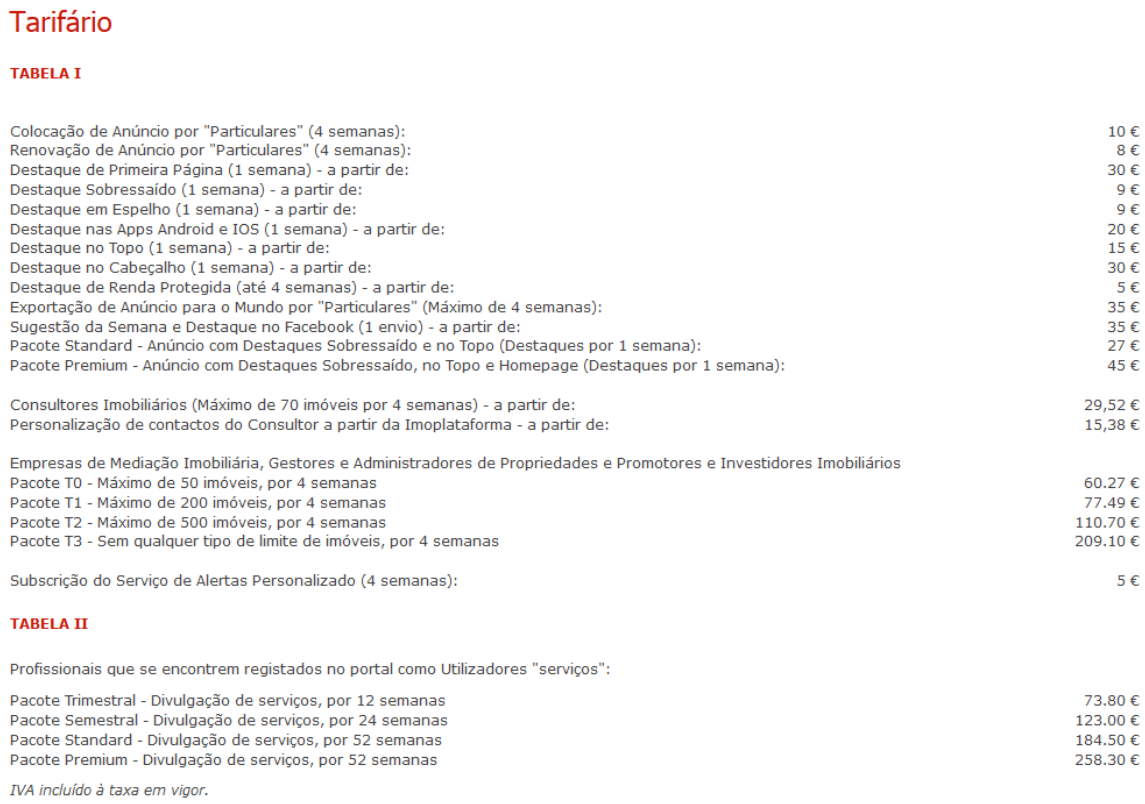


Figura 5 - Imovirtual, Pesquisa imóveis. [6]

Para se poder colocar anúncios no Imovirtual é necessário pagar um determinado valor dependendo das condições que queremos no anúncio, número de anúncios e ainda se for independente ou empresa como se pode ver na Figura 6 [7].



Tarifário

TABELA I

Colocação de Anúncio por "Particulares" (4 semanas):	10 €
Renovação de Anúncio por "Particulares" (4 semanas):	8 €
Destaque de Primeira Página (1 semana) - a partir de:	30 €
Destaque Sobressaído (1 semana) - a partir de:	9 €
Destaque em Espelho (1 semana) - a partir de:	9 €
Destaque nas Apps Android e IOS (1 semana) - a partir de:	20 €
Destaque no Topo (1 semana) - a partir de:	15 €
Destaque no Cabeçalho (1 semana) - a partir de:	30 €
Destaque de Renda Protegida (até 4 semanas) - a partir de:	5 €
Exportação de Anúncio para o Mundo por "Particulares" (Máximo de 4 semanas):	35 €
Sugestão da Semana e Destaque no Facebook (1 envio) - a partir de:	35 €
Pacote Standard - Anúncio com Destaques Sobressaído e no Topo (Destaques por 1 semana):	27 €
Pacote Premium - Anúncio com Destaques Sobressaído, no Topo e Homepage (Destaques por 1 semana):	45 €
Consultores Imobiliários (Máximo de 70 imóveis por 4 semanas) - a partir de:	29,52 €
Personalização de contactos do Consultor a partir da Imoplataforma - a partir de:	15,38 €
Empresas de Mediação Imobiliária, Gestores e Administradores de Propriedades e Promotores e Investidores Imobiliários	
Pacote T0 - Máximo de 50 imóveis, por 4 semanas	60.27 €
Pacote T1 - Máximo de 200 imóveis, por 4 semanas	77.49 €
Pacote T2 - Máximo de 500 imóveis, por 4 semanas	110.70 €
Pacote T3 - Sem qualquer tipo de limite de imóveis, por 4 semanas	209.10 €
Subscrição do Serviço de Alertas Personalizado (4 semanas):	5 €

TABELA II

Profissionais que se encontrem registados no portal como Utilizadores "serviços":

Pacote Trimestral - Divulgação de serviços, por 12 semanas	73.80 €
Pacote Semestral - Divulgação de serviços, por 24 semanas	123.00 €
Pacote Standard - Divulgação de serviços, por 52 semanas	184.50 €
Pacote Premium - Divulgação de serviços, por 52 semanas	258.30 €

IVA incluído à taxa em vigor.

Figura 6 - Imovirtual, Preços [7]

Comparativamente ao Uniplaces, a Imovirtual permite publicitar uma maior variedade de tipologias de imóveis, ainda assim, a solução que será desenvolvida pretende disponibilizar mais funcionalidades do que os serviços até agora apresentados.

2.2 SOLUÇÕES DE FATURAÇÃO ONLINE

Pretende-se integrar na solução uma oferta ao nível de sistema de faturação, para isso foi necessário procurar soluções de faturação *online* que permitissem o consumo dos seus serviços através de uma API. Neste capítulo iremos analisar quais os requisitos e características de algumas das soluções encontradas e fazer a comparação das mesmas, sendo que a que mais se enquadrar com os nossos objetivos será a utilizada no desenvolvimento do nosso serviço.

2.2.1 Requisitos

Um dos principais requisitos da nossa solução é a necessidade de um ou mais serviços de faturação, que possam ser integrados com a nossa solução. Para tal é fundamental que haja uma forma de interação entre esses serviços com a solução a ser desenvolvida. Por este motivo irão ser analisadas soluções de faturação que forneçam o acesso a uma API para o consumo dos seus serviços. A API terá de permitir que seja feita a faturação, emissão de faturas e recibos de cada entidade da nossa solução de forma independente. É ainda necessário que a API permita a inserção de novas entidades e que todas as ações possam ser realizadas sem que o utilizador tenha de interagir diretamente com a solução de faturação.

2.2.2 Soluções existentes

Nesta secção iremos analisar três serviços de faturação *online*: o Moloni, Invoicexpress, e o weoInvoice.

2.2.2.1 Moloni

O Moloni é um serviço *online* de gestão comercial. Permite o controlo da faturação, contas correntes, encomendas, *stocks*, entre outras funcionalidades. É um sistema certificado pela Autoridade Tributária e Aduaneira (identidade com a função de administrar os impostos, os direitos aduaneiros e os demais tributos em Portugal), simples e intuitivo. [8] O Moloni pode ser acedido através da simples utilização de um dispositivo com acesso à internet e um *browser*, mas oferece também uma API para que os seus serviços possam ser integrados em outros sistemas, este é um dos fatores pelo qual o Moloni se torna num dos serviços a considerar integrar na nossa solução [9].

Sendo este um sistema que oferece as suas funcionalidades como um serviço, permite que o consumo deste não tenha os preços elevados como outros sistemas que funcionam num

computador local, além disso, como a sua distribuição e comercialização é de *software* como serviço não existe necessidade de preocupação do lado do consumidor com atualizações. Posto isto, temos a garantia que estamos a trabalhar sempre com a versão mais recente deste *software* e também que este está sempre de acordo com a legislação da Autoridade Tributária e Aduaneira [8].

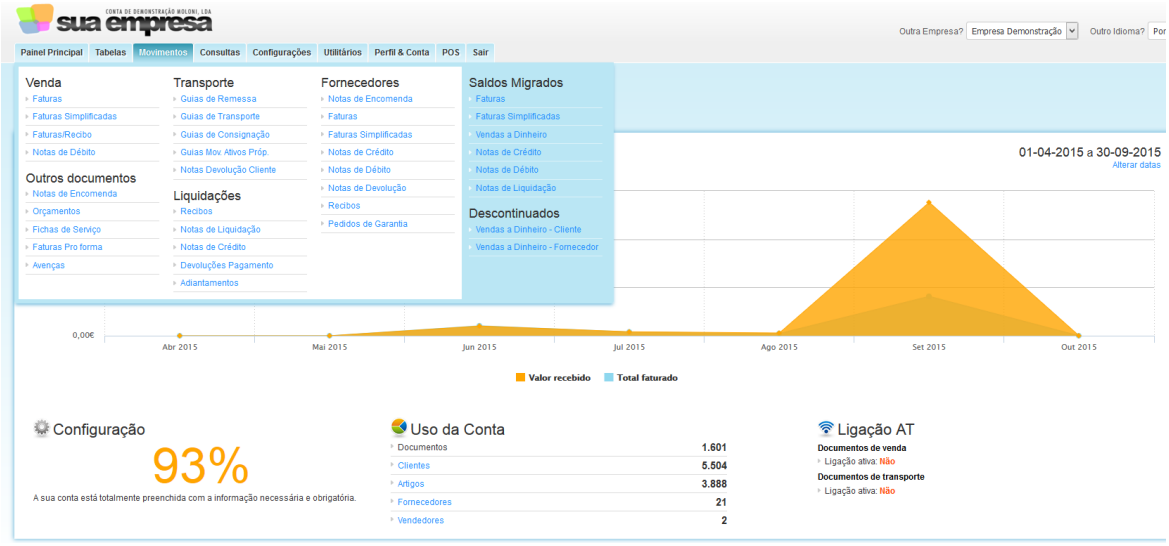


Figura 7 - Moloni, Página principal exemplo. [10]

Relativamente à área de desenvolvimento, o *Moloni* oferece uma *Sandbox* para que possa ser testada e haja assim uma melhor compreensão da sua API. Permite também que quem pretenda desenvolver usando a sua API não necessite de fazer qualquer pagamento [9].

2.2.2.2 INVOICEXPRESS

O Invoicexpress é outro dos serviços que nos permite fazer gestão comercial de negócios. Surgiu no mercado com o objetivo de tornar a criação e preenchimento de uma fatura o mais simples, rápido e fácil possível. Permite a criação e gestão de faturas, recibos, notas de crédito, encomendas, *stocks*, guias de transporte, orçamentos, estatísticas, entre outras funcionalidades [11].

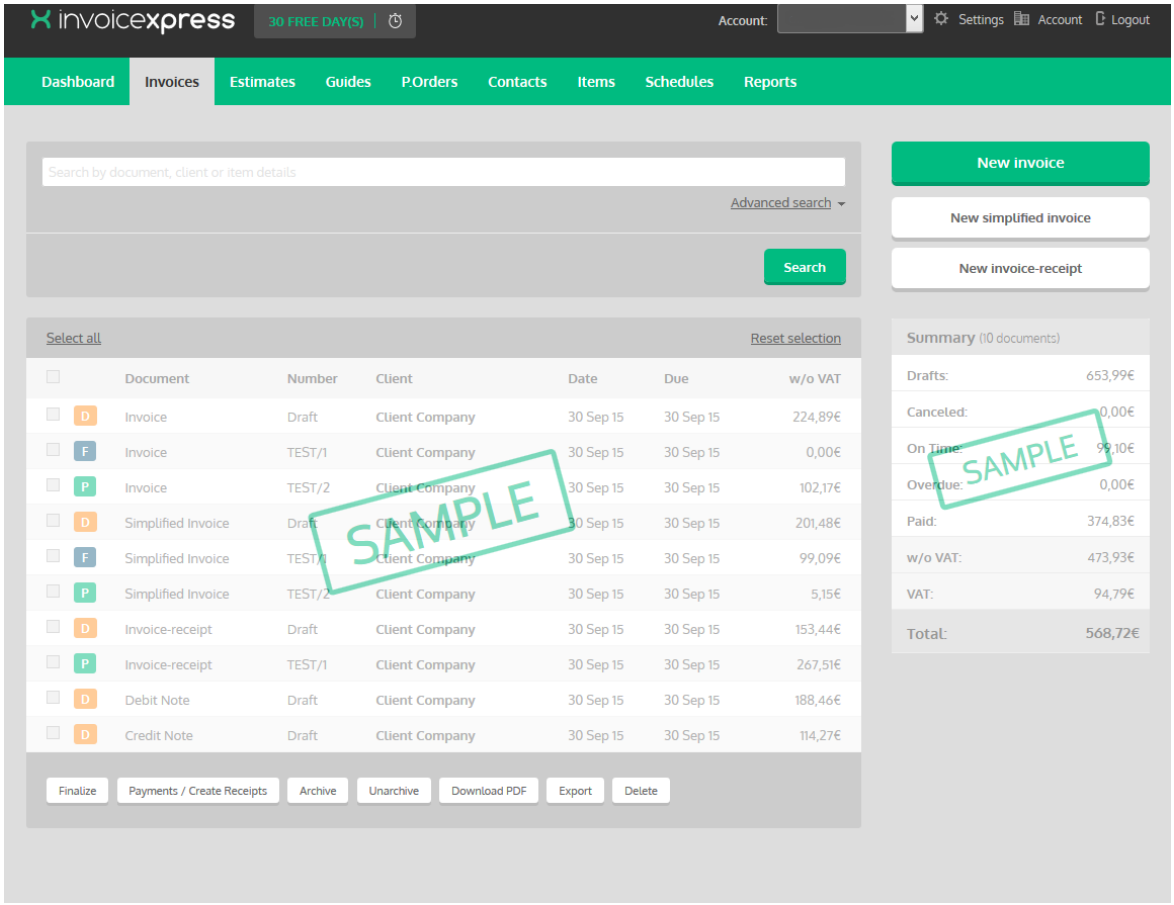


Figura 8 - Invoicexpress, Página principal exemplo.

Este serviço oferece também uma API para que possam ser integrados os seus serviços em outras soluções. É graças a esta API que o Invoicexpress se torna mais um possível candidato a ser integrado na nossa solução [12].

O Invoicexpress oferece as suas funcionalidades também como serviço. O consumidor tem à sua disposição uma diversidade de planos dos quais pode escolher o que mais se adequa às suas necessidades, assim não existe a necessidade do utilizador se preocupar com atualizações. Além disso, o Invoicexpress tem o seu sistema certificado pela Autoridade Tributária e Aduaneira, e ainda garante que ao utilizar o seu serviço estaremos sempre atualizados e em conformidade com a legislação em vigor [11].

Um ponto que o Invoicexpress oferece comparativamente a outros serviços é a disponibilidade de aplicações móveis para os principais sistemas operativos móveis, Android, iPhone(iOS) e Windows Phone [12].

2.2.2.3 *WeoInvoice*

O weoInvoice é mais um serviço um serviço de faturação *online*. Este serviço tem uma característica única em relação aos outros analisados que é a existência de um plano completamente gratuito sem limite de faturas. Este serviço permite fazer gestão de faturas, produtos, clientes e fornecedores. Apesar da grande vantagem de ser um serviço com um plano gratuito, este serviço peca pela falta de abertura em relação a sua API. A única informação disponível é que a utilização da API é feita sobre consulta [13].

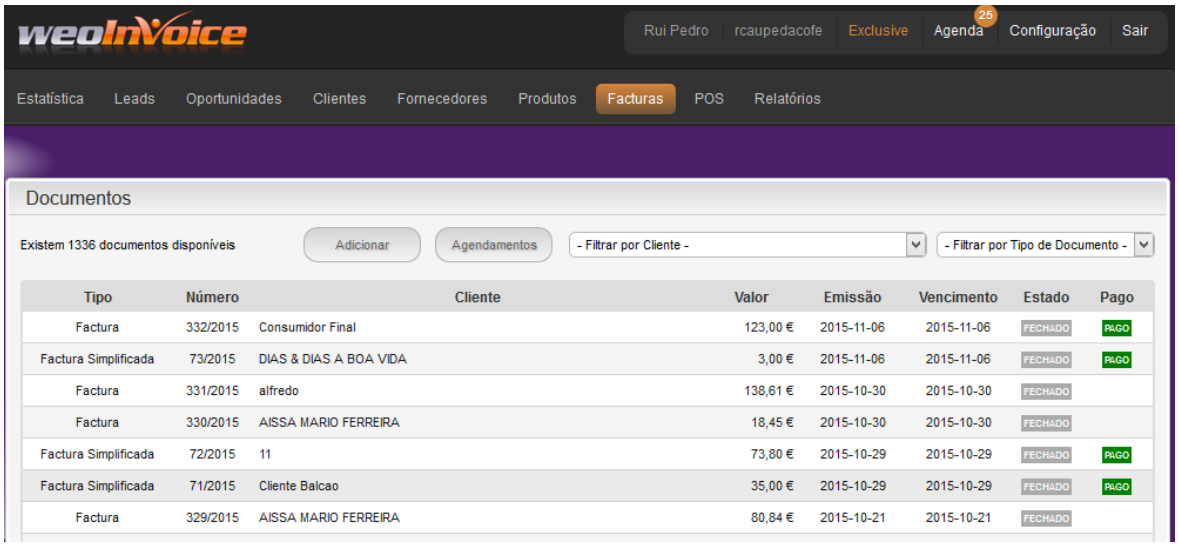


Figura 9 - weoInvoice, Página de faturas exemplo.

2.2.3 Comparações

Foram analisados anteriormente três serviços de faturação diferentes. Ambos os serviços permitem fazer a faturação que pretendemos no nosso sistema mas existem vantagens e desvantagens em ambos. Estes serviços já estão certificados com a AT e já cumprem as leis necessárias para a realização da faturação. Isto trás uma enorme vantagem para nós, tendo em conta que o nosso objetivo não é apenas fazer a faturação. Utilizar estes serviços facilita imenso o processo de inserção de faturação no nosso sistema, pois grande parte da responsabilidade passa para esses serviços. Estes têm de estar sempre atualizados com a lei o que torna esta uma das principais vantagens da utilização dos mesmo. Além disso a

responsabilidade de garantir a comunicação e certificação com a AT é assumida por esses serviços. A principal diferença entre eles são os custos. Apesar do weoInvoice ser o único que dispõe de um plano gratuito, devido à falta de informação em relação à API, não foi escolhido para ser o primeiro serviço de faturação a ser integrado no nosso sistema.

Tanto o Moloni como o INVOICEXPRESS envolvem custos na sua utilização e, apesar desses valores serem diferentes, o Moloni traz algumas vantagens com custos ligeiramente mais baixos. Uma dessas vantagens é o limite de documentos mensais, onde o Moloni não oferece limite [14] [15]. Além disso, e um dos principais fatores que levou à escolha do Moloni em vez do InvoiceXpress, foi a disponibilização de uma *sandbox*. Esta permite experimentar os pedidos à API e facilita a compreensão do funcionamento dos mesmos. Estes foram os motivos que nos levaram à escolha do Moloni como o primeiro serviço de faturação a ser inserido no nosso sistema

2.3 LEGISLAÇÃO DE ARRENDAMENTOS PARA PESSOAS SINGULARES

Em 1 de Janeiro de 2015 entrou em vigor a lei n.º 82-B/2014 e n.º 82-E/2014, de 31 de dezembro. Estas leis introduziram a obrigatoriedade, por parte das pessoas singulares, de comunicação dos contratos de arrendamento à AT, e também a emissão de recibo de quitação eletrónico [16] [17] [18]. Tanto a comunicação dos contratos como a emissão dos recibos é feita obrigatoriamente através do portal das finanças ou usando a API disponível através de serviços externos [16]. Esta API foi disponibilizada pela AT a partir de 31 de Julho de 2015. A implementação desta API no nosso sistema é de extremo interesse para os nossos clientes que se encontrarem na categoria de pessoas singulares, pois irá permitir que estes possam realizar também todo o processo de arrendamento através da nossa solução. Não existe nenhum custo associado ao consumo desta API, apenas é necessário obter um certificado SSL assinado pela AT. Infelizmente, como este serviço apenas apareceu no final de julho, não houve tempo para a sua implementação neste serviço [19].

3. CONCEÇÃO E DESENVOLVIMENTO

3.1 REQUISITOS

Nesta dissertação tem-se como principal objetivo o desenvolvimento de uma solução que permita a uma ou mais entidades fazerem todo o processo de gestão de arrendamentos dos seus imóveis. Para isso, será desenvolvida uma aplicação web que irá fornecer um serviço que permita a um conjunto de entidades gerir os seus imóveis, publicitar imóveis disponíveis para arrendamento, gerir os seus contratos e gerir os seus pagamentos e faturação. Para ser possível analisar mais facilmente os requisitos de uma aplicação que tenha as características acima referidas, iremos separar os requisitos através de quatro pontos: a gestão dos utilizadores e os seus cargos, gestão dos imóveis, gestão de contratos e por fim gestão de pagamentos.

3.1.1 Gestão de utilizadores

A aplicação que se pretende desenvolver irá fornecer um serviço a um conjunto de entidades. Para fornecer esse serviço, irá necessitar de uma equipa para gerir a aplicação, para isso será necessário criar ferramentas que permitam gerir os utilizadores da aplicação. Por sua vez a aplicação terá de permitir a várias entidades, desde empresas a pessoas singulares, utilizar simultaneamente os mesmos recursos do serviço mantendo sempre essas entidades separadas. Este conceito tem o nome de *multi-tenant* [20]. Isto significa que será possível na mesma infraestrutura suportar inúmeras entidades na mesma instância de *software* trazendo assim vantagens como custos operacionais menores e facilidade de gestão de *upgrades*, pois o fornecedor de serviço apenas tem de gerir uma instância para todos os seus clientes ao invés de uma por cliente. Por fim, será necessário existir um nível para os clientes das entidades que utilizarão o serviço, estes serão os

interessados em apenas arrendar imóveis. Na figura abaixo podemos ver os três níveis de utilizadores e as relações entre eles.

Figura 10 - Níveis de utilizadores e suas relações

3.1.2 Gestão de Imóveis

O segundo ponto dos requisitos é a gestão dos imóveis e dos três níveis de utilizadores. O único nível de utilizador que não irá ter nenhuma relação com a gestão de imóveis será o fornecedor do serviço. A aplicação terá de permitir às entidades de arrendamentos, a possibilidade de fazer a gestão de vários imóveis, ou seja, será necessário permitir a inserção de novos imóveis, fazer a manutenção dos dados associados, poder editar ou remover imóveis e, no caso de por exemplo empresas mediadoras, gerir os seus proprietários. Sendo que o objetivo é arrendar os imóveis, será necessário que a entidade de arrendamentos possa publicitar os seus imóveis. Por este motivo, terá de existir um local onde os inquilinos consigam pesquisar e ver as suas características. Na imagem a baixo podemos ver o fluxo desde a inserção do imóvel até ao início do contrato.

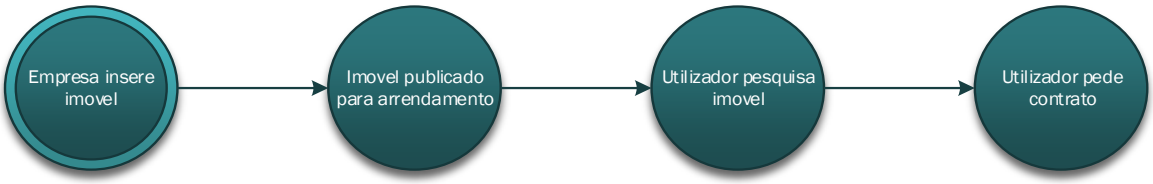


Figura 11 - Inicio fluxo imóvel

3.1.3 Gestão de contratos

Chegamos assim ao terceiro ponto dos requisitos, a gestão de contratos. Depois do inquilino fazer um pedido de contrato, a entidade gestora do imóvel e este, terão de ter as ferramentas necessárias para comunicarem e trocarem os documentos necessários à realização do contrato. Na imagem seguinte podemos ver o fluxo básico das fases que a aplicação terá de suportar para que um contrato possa ser aceite e seja dado o início do mesmo.

Figura 12 - Fluxo para início de contrato

3.1.4 Gestão de Pagamentos

Por fim, e já em relação à gestão de pagamentos, quando um contrato é aceite será necessário que tanto o inquilino como a entidade gestora do imóvel possam ver e, no caso do inquilino, realizar ao longo do tempo os pagamentos relacionados com esse contrato. Para isso, cada contrato terá toda a gestão dos seus pagamentos individualizada sem que haja cruzamento de informação de pagamentos com outros contratos. Consoante o tipo de entidade de arrendamentos, para cada pagamento poderá ser necessário emitir a sua respetiva fatura ou recibo que terão de estar disponíveis para *download* juntamente com a informação do pagamento.

Será ainda necessário existir uma zona onde as entidades de arrendamentos possam realizar os pagamentos pelo consumo do serviço. Cada uma delas terá os seus pagamentos individualizados e apenas pagará consoante o que consumiu.

3.2 ARQUITETURA

Na nossa arquitetura um dos pontos a que foi dada especial atenção foi o suporte à capacidade de integração de novos serviços de faturação. Pretende-se que a faturação no nosso sistema seja o mais dinâmica possível, ou seja, pretende-se que as entidades de arrendamentos que façam faturação não estejam sujeitas apenas a um serviço de faturação, mas sim que a um conjunto de serviços com a possibilidade de, caso seja pretendido, se consiga facilmente adicionar novos serviços de faturação. Por este motivo, a arquitetura da solução tem de permitir que novos serviços externos de faturação sejam inseridos sem que seja necessário alterar o *core* da aplicação.

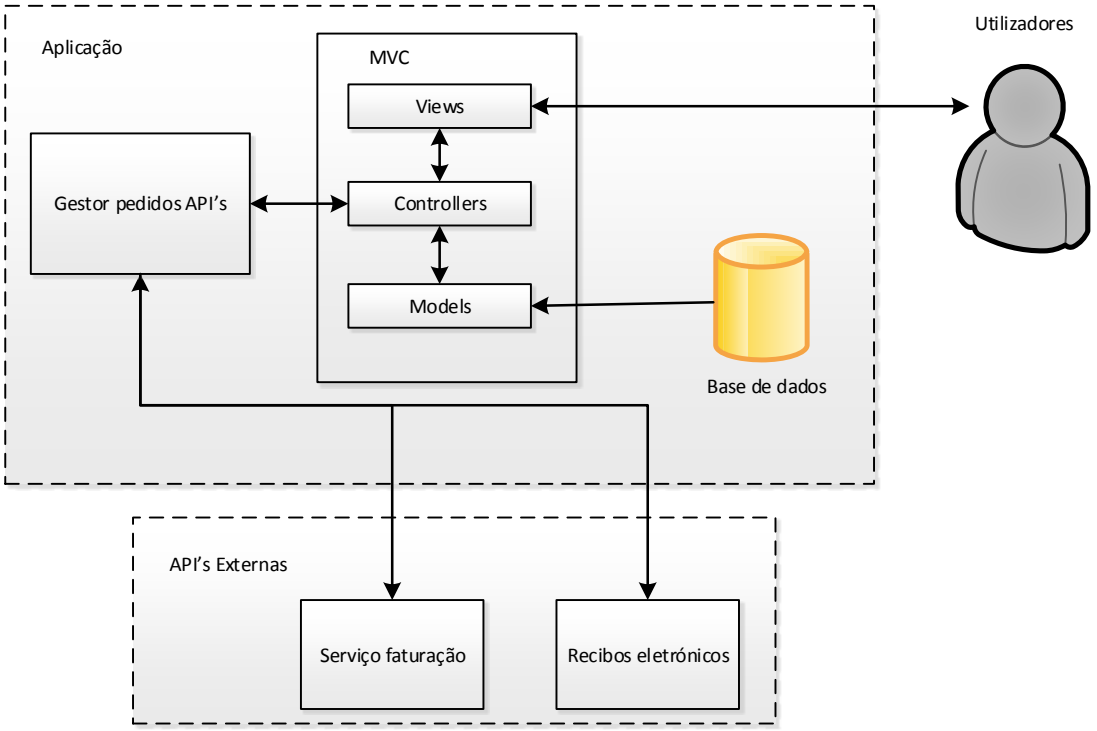


Figura 13 - Arquitetura sistema

Assim como podemos ver na Figura 13, a arquitetura do sistema é composta por duas partes principais, a aplicação que será desenvolvida e as APIs externas que serão consumidas. A aplicação vai ser constituída por três principais componentes, um

componente MVC, uma base de dados e um componente de gestão dos pedidos as APIs. O componente MVC é o principal componente de toda a aplicação e é neste que será definida toda a lógica da mesma. Este componente estará dividido em três grandes partes, os *Models* que será onde irá estar a informação da estruturação dos dados na aplicação, os *Controllers* que será onde irá ser definida a lógica da aplicação e que fará ligação entre os *Models* e as *Views* e por fim as *Views* que será o componente que irá fazer a representação da informação para o utilizador e enviar os pedidos do mesmo para os *Controllers*. A aplicação será composta também por uma base de dados que será responsável por guardar os dados da aplicação. O último componente da aplicação será um gestor de pedidos a serem realizados às APIs. Como já foi referido, o objetivo é que a aplicação seja dinâmica em relação a utilização das APIs, por esse motivo, o gestor de pedidos irá ter a função de isolar as chamadas às APIs do *core* da aplicação. Assim quando for necessário fazer uma chamada a uma API, o *Controller* apenas terá de indicar ao gestor de pedidos qual a API que pretende realizar o pedido.

3.3 MODELO DE DADOS

No modelo de dados é onde se vai definir o modo como estão estruturados os dados a armazenar na base de dados e as relações entre eles. Assim, criou-se um modelo que tem como objetivo ser capaz de suportar toda a informação necessária para a aplicação. Na Figura 33 (em anexo) está disponível o diagrama do modelo completo. Neste capítulo irão ser analisadas as tabelas principais do nosso modelo.

O primeiro grupo de tabelas a analisar diz respeito à tabela **Imovel** e as tabelas a ela associadas.

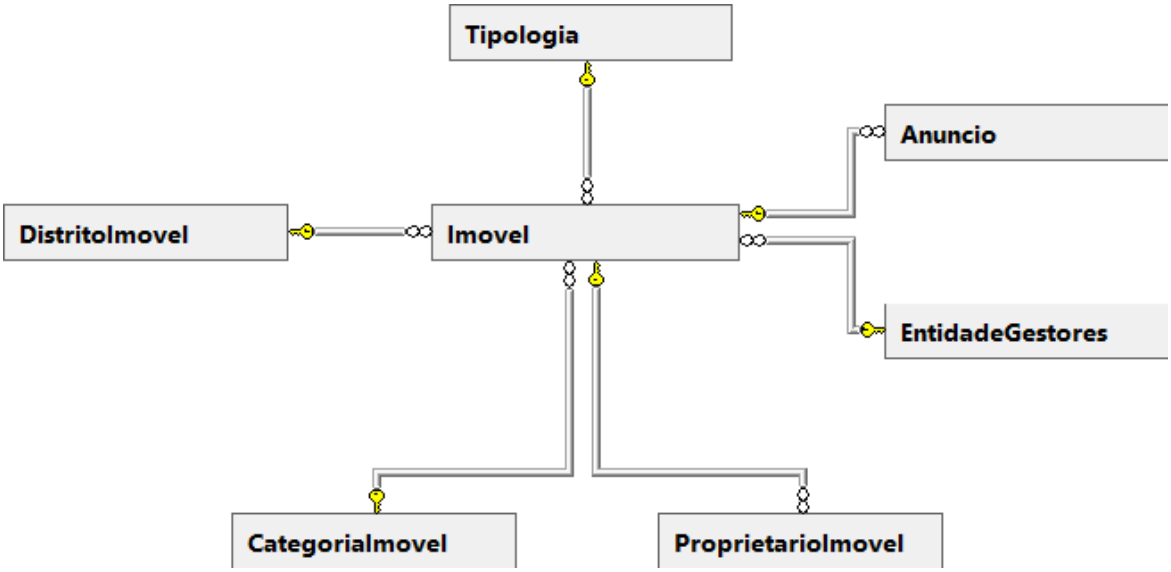


Figura 14 - Tabela imóvel e seus principais adjacentes

A tabela **Imovel** é uma das principais do sistema. Esta é onde serão armazenados os dados dos imóveis que irão ser inseridos no sistema. É composta pelos seguintes pontos:

Atributo	Descrição
ImovelCodigo	Identificador único de cada imóvel da tabela
Morada	Morada do imóvel
ValorArrendamentoBase	Valor da renda do imóvel
ValorDespesasBase	Valor das despesas do imóvel
EntidadeGestoresId	Identificador único da entidade gestora do imóvel
CategoriaImovelId	Identificador único da categoria do imóvel
DistritoId	Identificador único do distrito do imóvel
Latitude	Latitude do imóvel
Longitude	Longitude do imóvel
TipologiaId	Identificador único da tipologia do imóvel
SGImovelId	Identificador único do imóvel no serviço de faturação

Tabela 1 - Atributos tabela Imovel

Como se pode ver na Figura 14, a tabela imóvel está ligada com as tabelas **Tipologia**, **DistritoImovel**, **CategoriaImovel**, **Anuncio**, **EntidadeGestores** e **ProprietarioImovel**. As tabelas **Tipologia**, **DistritoImovel** e **CategoriaImovel** tal como o nome indica será onde irão estar guardadas as diferentes tipologias de imóveis, diferentes distritos e as possíveis categorias. Cada imóvel irá ter uma tipologia, um distrito e uma categoria, como por exemplo T0, Aveiro e Apartamento. A tabela **Imovel** está também ligada à tabela **Anuncio**. Cada imóvel da tabela **Anuncio** poderá ter diversos anúncios, pois um imóvel pode ficar disponível para arrendar diversas vezes, logo poderá ter vários anúncios. Por fim a tabela **Imovel** está ainda ligada às tabelas **EntidadeGestores** e **ProprietarioImovel**. A tabela **EntidadeGestores** será onde irão ficar guardadas as entidades de arrendamento e cada imóvel poderá ter apenas uma entidade de arrendamento. A **ProprietarioImovel** é onde irão ficar guardados os proprietários de imóveis. Cada imóvel é gerido apenas pela sua entidade de arrendamento mas isso não significa que essa entidade seja a proprietária do imóvel, ou até a única, por isso cada elemento da tabela **Imovel** poderá ter vários proprietários na tabela **ProprietarioImovel**.

Para poder ser arrendado um imóvel é necessário um contrato, por este motivo vamos analisar a tabela **Contrato** e as suas tabelas associadas. Esta tabela é também de grande importância para o sistema e tem como função armazenar os dados relativos aos contratos dos imóveis. Na tabela abaixo podemos ver os seus atributos.

Atributo	**Descrição**
ContratoID	Identificador único de cada contrato da tabela
InquilinoID	Identificador único do inquilino do contrato
ImovelCodigo	Identificador único do imóvel do contrato
DataInicioContrato	Data de início do contrato
DataFimContrato	Data de fim do contrato
ValorArrendamento	Valor da renda do contrato
ValorDespesas	Valor das despesas do contrato
SemRetencao	Se o contrato tem retenção

ProprietarioID	Identificador único do proprietário titular do imóvel
Observacoes	Observações do contrato
DataPedidoContrato	Data em que o contrato foi pedido
Estado	Estado em que o contrato se encontra

Tabela 2 - Atributos tabela Contrato

Na Figura 15 podemos ver a tabela Contrato e as suas associadas que são a tabela **Facturas**, **Imovel**, **GestaoMonetaria**, **Inquilino**, **DocumentosContratos**, **Pagamentos**, **Proprietario** e **Recibo**. Existem diversas variações da composição de um contrato. Este pode ser composto por vários inquilinos e vários proprietários, mas de modo a simplificar o desenvolvimento do protótipo foi decidido que o contrato iria apenas ser composto por um inquilino e um proprietário titular. Assim sendo, cada contrato poderá então ter um imóvel, um inquilino e um proprietário (titular) e essas relações são representadas pelas ligações entre a tabela **Contrato** e as tabelas **Imovel**, **Inquilino e Proprietario.** Como para existir um contrato é necessário um conjunto de documentos será usada a tabela **DocumentosContratos** para esse efeito. Para a gestão de pagamentos do contrato é usada a tabela **Pagamentos** e a tabela **GestaoMonetaria**. Na tabela **Pagamentos** irão ser armazenados todos os pagamentos, saldados ou por saldar, associados ao contrato. A tabela **GestaoMonetaria** será apenas usada para gerir a dívida total do inquilino em relação ao contrato. Por fim, associada à tabela **Contrato**, a tabela **Facturas** e **Recibo** serão usadas para auxiliar, caso a entidade pretenda fazer faturação no sistema, no armazenamento dos dados sobre emissão de faturas e recibos. Apesar de se ir usar serviços externos para faturação é necessário a existências destas duas tabelas para gestão e estruturação geral dessa informação no sistema.

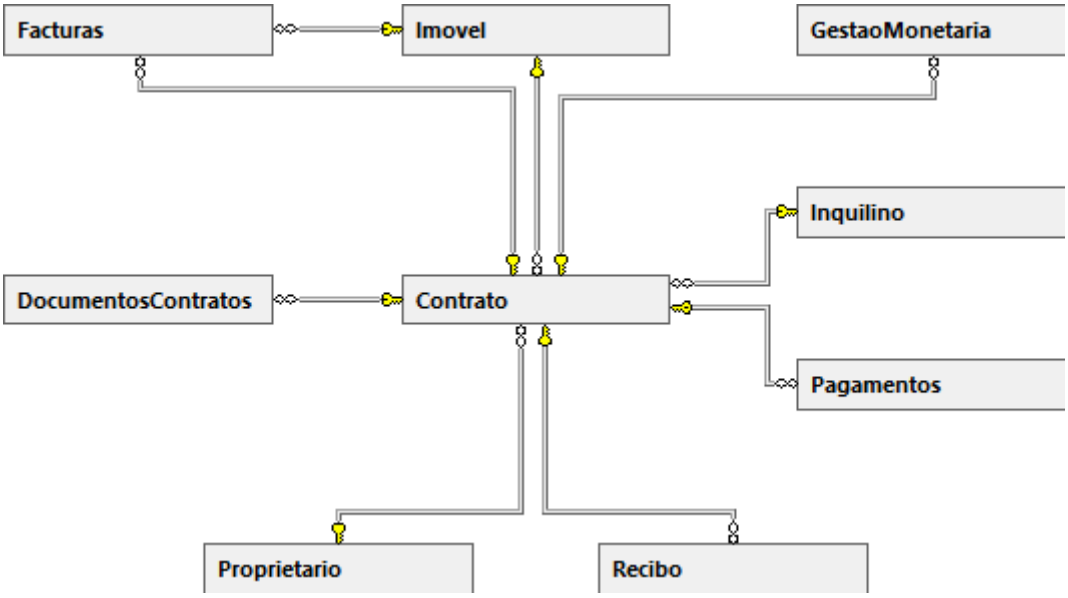


Figura 15 - Tabela Contrato e seus principais adjacentes

O último grupo de tabelas é composto pela tabela **Entidade** e as tabelas relacionadas com ela. Como se pode ver abaixo na Figura 16 a tabela **Entidade** está relacionada com a tabela **EntidadeGestores** e também a tabela **Inquilino**. Isto deve-se ao facto de todos os utilizadores da aplicação, sejam eles empresas ou inquilinos, partilharem as mesmas características básicas como por exemplo o nome e as informações de contacto.

Atributo	Descrição
ContactoTelefonico	Contacto telefónico da entidade
Nome	Nome da entidade
NumContribuinte	Número de contribuinte da entidade
ContactoEmail	Correio eletrónico da entidade
AspNetUserId	Identificador único da informação de login da entidade
EntidadeId	Identificador único de cada entidade da tabela
Morada	Morada da entidade
EntidadeGestoresId	Identificador único da entidade gestora a que esta entidade pertence

Tabela 3 - Atributos tabela Entidade

Caso o utilizador seja inquilino, a tabela **Entidade** apenas se irá relacionar com a tabela **Inquilino**. Por outro lado, se for uma entidade de arrendamento, a tabela **Entidade** já se irá relacionar com as tabelas **DefiniçoesEntidade**, **EntidadeGestores** e **PagamentosEntidade**. A tabela **DefiniçoesEntidade** é responsável por armazenar definições acerca da entidade à qual está relacionada. Essas definições podem ser referentes à utilização, ou não, dos serviços de faturação integrados no sistema ou definições de modelos de pagamento. Na tabela **EntidadeGestores** são guardadas as informações das entidades de arrendamentos. Estas podem ser compostas por vários membros associados (tabela **Entidade**), mas apenas um, com a função de administrador (tabela **AspNetRoles**), terá acesso às definições e às informações dos pagamentos ao sistema (tabela **PagamentosEntidade**). Isto leva-nos às restantes tabelas que são **AspNetUsers**, **AspNetUserRoles** e **AspNetRoles** que será onde as informações de *login* e as *roles* de cada entidade irão ser armazenadas.

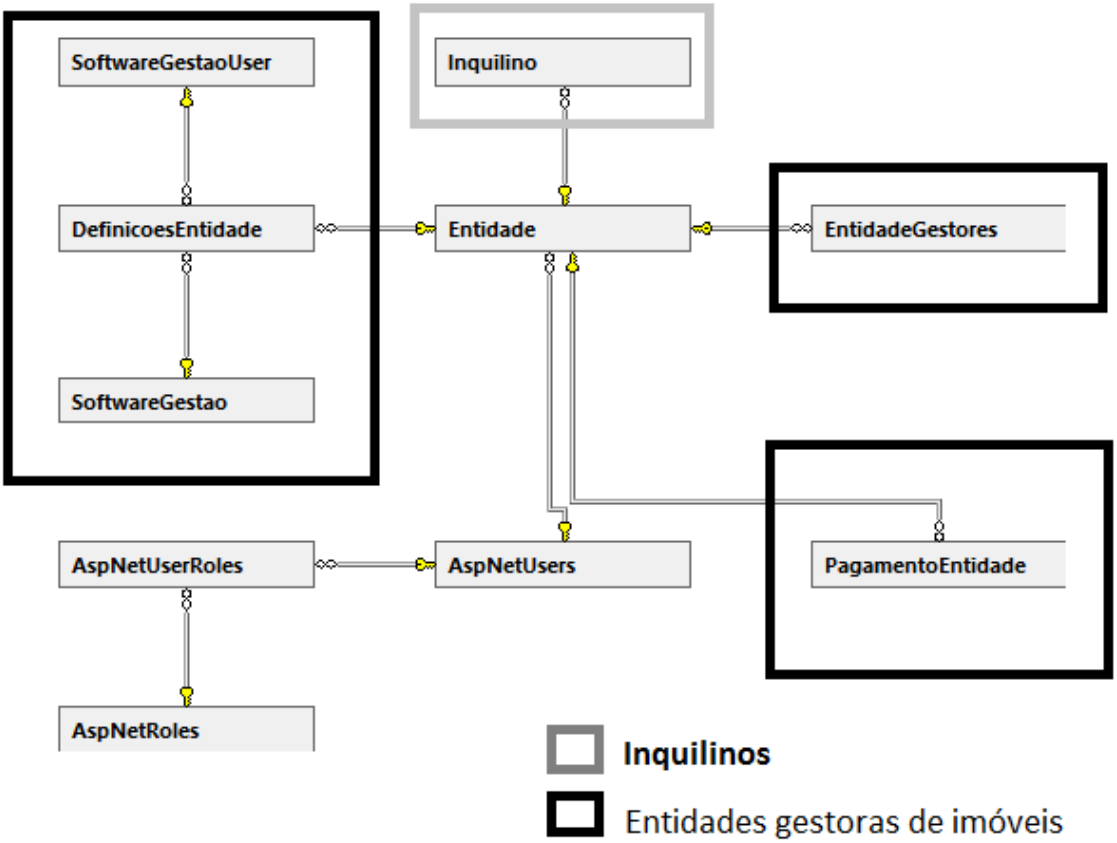


Figura 16 - Tabela Entidade e adjacentes

3.4 IMPLEMENTAÇÃO

Neste subcapítulo será explicado como foram implementados os componentes da aplicação. Para a implementação dos componentes que serão explicados em baixo, foram usadas várias tecnologias que podem ser vistas no anexo 7.1.

3.4.1 Asp.Net MVC

Como já foi referido no subcapítulo da arquitetura do sistema, este componente é o de maior peso em toda a aplicação. Este está dividido em três grandes partes, *Models*, *Controllers* e *Views*. Para iniciar a implementação deste componente usou-se uma ferramenta chamada *Entity framework* que permite a criação automática dos *Models*, ou modelos, usando uma base de dados já implementada. Para isto, foi primeiro implementada a base de dados e depois, usando esta ferramenta, foram gerados os modelos deste componente. O resultado é um conjunto de modelos que definem as propriedades das tabelas na base de dados e ficheiros de configuração que permitem ao sistema saber como comunicar com a base de dados. Com isto feito, tornou-se possível fazer pedidos à base de dados através dos *Controllers*. Em baixo podemos ver um exemplo de um pedido à base de dados:

```
private AppEntity db = new AppEntity();
Entidade entidade = db.Entidade.Where(i => i.AspNetUserId == user.Id).FirstOrDefault();
```

Com acesso à base de dados nos *Controllers* foram utilizados dois métodos para envio de informação para as *Views*, através de modelos e através de *ViewBags*. Os modelos são classes que representam uma estrutura de dados que é utilizada para transportar a informação até às *Views* para que seja mostrada aos utilizadores. As *ViewsBags* têm também a função de fazer chegar informação às *Views* mas apenas permite, por cada *ViewBag*, o envio de um tipo de dados sem verificar a estrutura dos mesmos, ou seja em situações de *Views* mais complexas é criado um modelo só para o transporte de dados dessa *View*, mas em situações de *Views* mais simples são utilizados *ViewBags* pois ajudam

a reduzir a quantidade de modelos na aplicação. Em baixo temos um exemplo onde se pode perceber a diferença entre a capacidade de um modelo e a capacidade de uma *ViewBag*.

```
public class ImovelInfoViewModel
    {
        public List<ImagemImovel> imgimovel { get; set; }
        public Imovel imovel { get; set; }
        public Anuncio anuncio { get; set; }
    }
```

O modelo acima permite transportar uma lista de imagens, todas as propriedades de um imóvel e ainda as propriedades do seu anúncio, garantido sempre a estrutura dos dados. Em baixo temos um *ViewBag* que apenas permite transportar uma lista de distritos.

```
ViewBag.Distrito = new SelectList(db.DistritoImovel.ToList(), "DistritoId", "Distrito");
```

Outra vantagem do uso de modelos é que estes podem ser usados para retornar novamente informação para o *Controller* para que sejam realizadas todas as operações de lógica necessárias.

3.4.2 Gestor Pedidos API

Para a realização de faturação recorrendo a serviços externos foi criado um componente que é responsável pela execução dos pedidos às APIs externas. Este componente comporta-se como uma biblioteca a que os *Controllers* podem recorrer sempre que necessitarem de comunicar com as APIs externas. Para cada API externa é criada uma classe (informações detalhadas no anexo 7.4) onde cada função representa um pedido diferente à API. Além disso, é criado um modelo que traduz as respostas enviadas pela API em estruturas de dados que podem ser percebidas pelos *Controllers*. Em baixo podemos ver um exemplo de como o *Controller* pode dinamicamente fazer o pedido a uma API:

```
switch (APIExterna)
{
        case "Moloni":
        MoloniResponses.Exemplo result = GestorPedidos.Moloni.ExemploPedido(…);
        break;

        case "OutraAPI":
        OutraAPIResponses.Exemplo result = GestorPedidos.OutraAPI.ExemploPedido(…);

        break;
}
```

Como se pode ver no trecho de código acima, o *Controller* apenas tem de saber a quem fazer o pedido ficando assim abstraído de como são realizados os pedidos. Deste modo, quando for necessário adicionar novas APIs ao componente MVC, este estará pronto a utilizá-las facilmente.

4. PROTÓTIPO

Neste capítulo vamos demonstrar o funcionamento do protótipo. Para isso, serão demonstradas todas as fases e ações a serem realizadas entre a entidade de arrendamento e um inquilino, desde o registo na aplicação até à finalização de um contrato.

4.1 GESTÃO UTILIZADORES

Para começar a utilizar a aplicação, tanto entidades de arrendamento como inquilinos terão de fazer o registo. Na Figura 17 podemos ver os campos que compõem o registo. A página é a mesma seja qual for o tipo de utilizador, mas apenas preenchem os últimos dois campos os utilizadores que pretendam gerir e arrendar os seus imóveis.

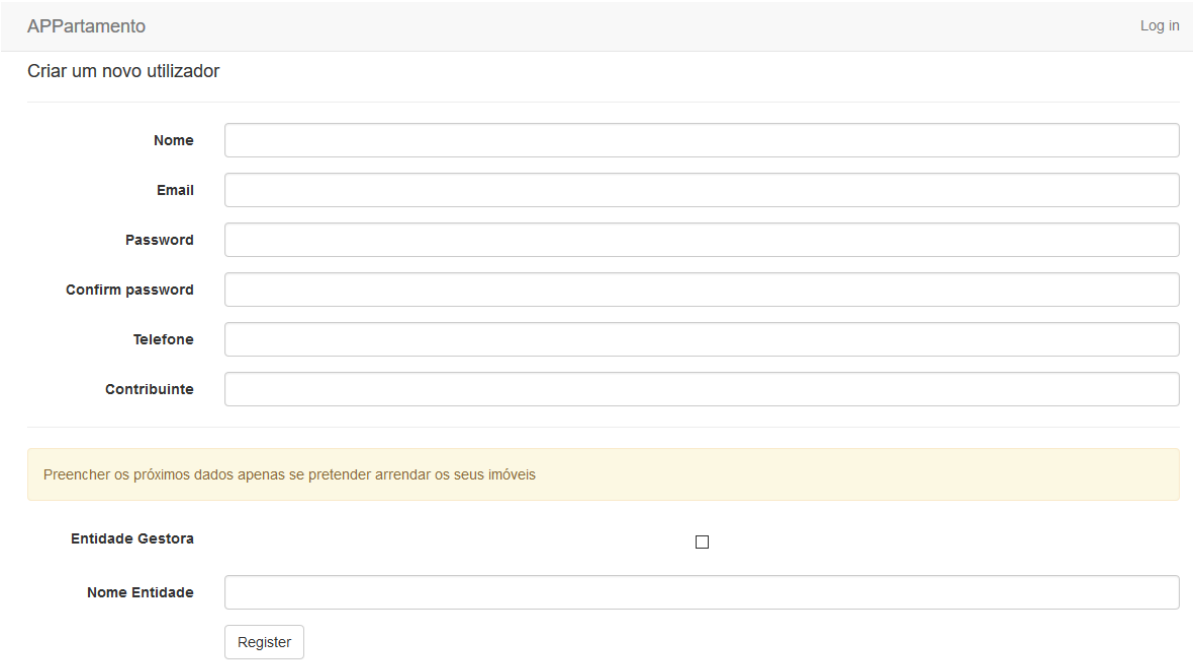


Figura 17 - Página de registo

A página inicial é também comum para as entidades de arrendamento e para os inquilinos. Esta é composta por três filtros e um botão de pesquisa. Assim, um utilizador pode pesquisar todos os imóveis sem filtros, ou filtrar consoante três características: a categoria (se é um apartamento ou moradia por exemplo), o distrito, e a tipologias (T1, T2, T3, entre outros). Esta página tem ainda um botão de *login* que permite aos utilizadores entrarem na aplicação. Na Figura 18 podemos ver a página inicial.

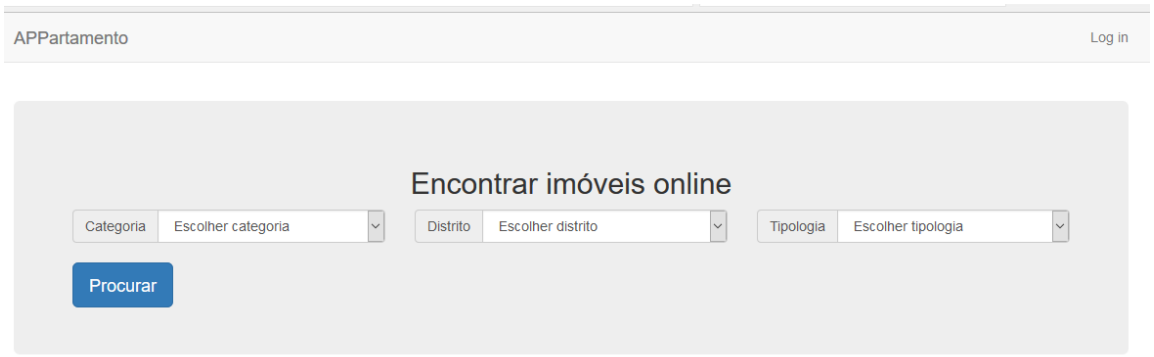


Figura 18 - Pagina inicial

Para fazer qualquer tipo de ação sem ser pesquisar imóveis é necessário os utilizadores fazerem *login* na aplicação. Assim, foi criada uma página de *login* que permite aos utilizadores acederem às suas contas. Como podemos ver na Figura 19, esta página é apenas composta por um campo de correio eletrónico, um de palavra-chave e o botão de login. Existe ainda uma opção de guardar os dados para facilitar o login num futuro acesso do utilizador à aplicação e ainda um botão de registo caso o utilizador ainda não tenha conta.

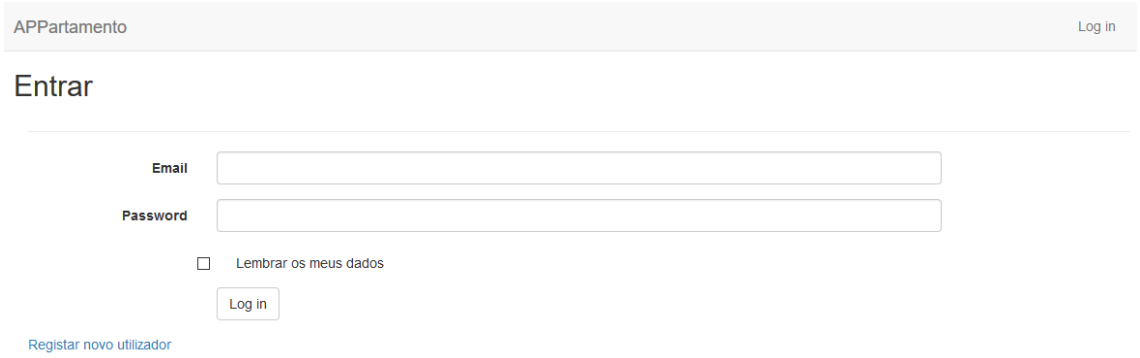


Figura 19 - Página login

4.2 GESTOR IMÓVEIS

Depois de uma entidade de arrendamento fazer login pela primeira vez, terá de começar por inserir imóveis na aplicação. Na Figura 20 é apresentada a página de criação de imóveis, onde são pedidas todas as informações necessárias para criação do imóvel. O título, a descrição e as imagens são os campos direcionados para o anúncio do imóvel, e as restantes, como a morada, valor renda e despesas, categoria, tipologia, distritos e local são apenas para definir as características do imóvel.

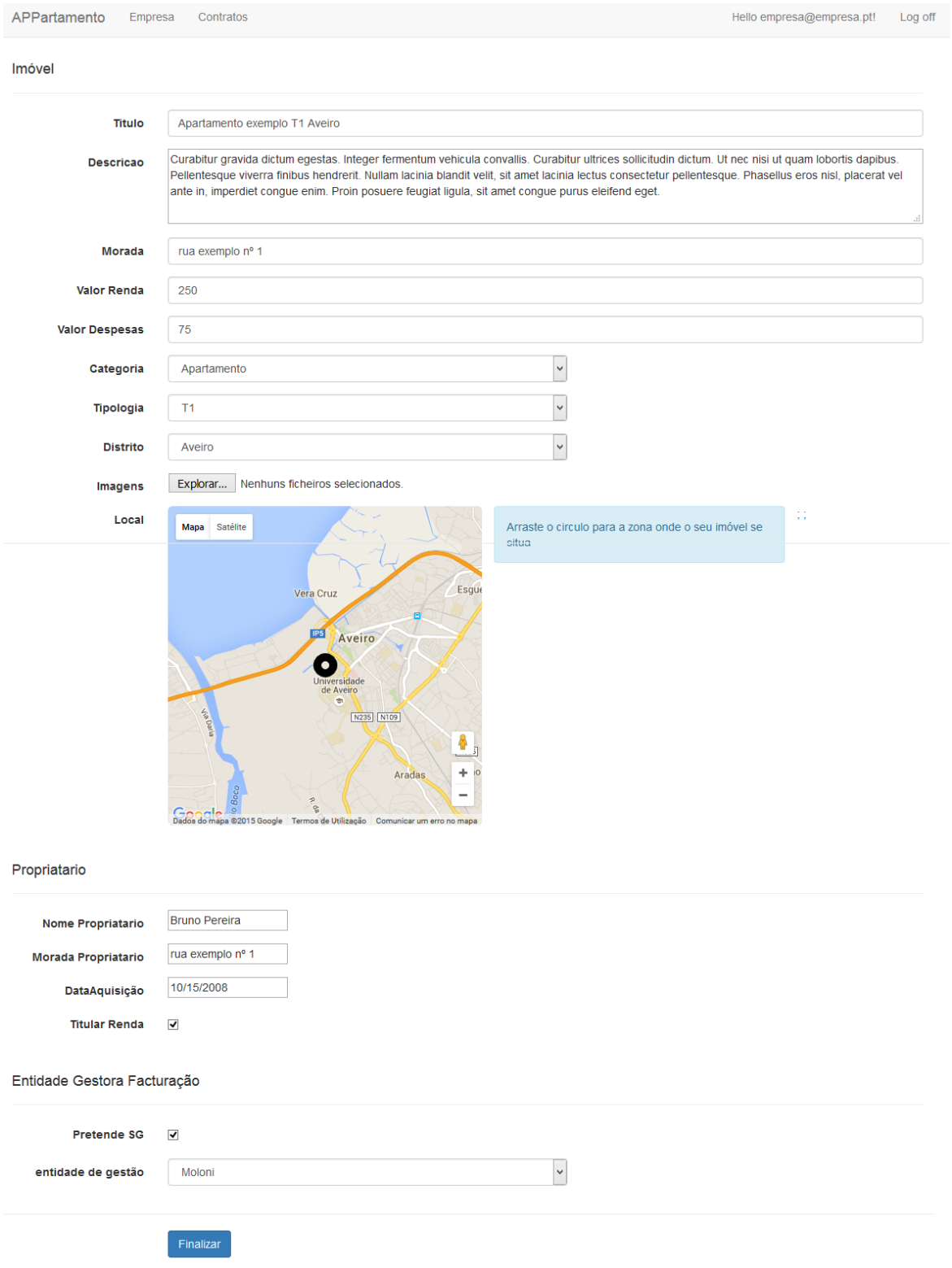


Figura 20 - Criar imóvel

O campo local é um mapa onde o utilizador pode marcar a posição do seu imóvel arrastando um ícone. Isto irá tornar mais fácil a identificação da localização do imóvel para os utilizadores que visualizarem o anúncio. De seguida, são pedidas as informações do proprietário do imóvel e por fim, caso ainda não tenha sido definido anteriormente, se a entidade de arrendamento pretende fazer faturação dos seus imóveis na aplicação.

Ao carregar no botão finalizar, a entidade de arrendamento será redirecionada para a página Empresa. Esta página permite que os gestores visualizem os seus imóveis, façam alterações às características dos seus imóveis e também gerir os proprietários desses imóveis.

Figura 21 - Página de gestão de imóveis

Do lado do futuro inquilino, a partir do momento em que existem imóveis no sistema, é possível obter resultados na pesquisa de imóveis da página inicial. Na Figura 22, em baixo, podemos ver um exemplo de resultado. A página é dividida a meio, no lado esquerdo existe um mapa com a sinalização da localização dos imóveis que foram encontrados e à direita podemos ver esses vários imóveis, com as suas imagens e algumas informações. Para abrir mais informações de um imóvel o utilizador clica no botão “Ver” do imóvel que pretende visualizar e será aberta uma nova página (Figura 23) que permite ver as imagens do imóvel, a descrição e as suas características.

Ao lado dessas informações o utilizador terá disponível um botão que dá acesso à página de pedido de contrato, botão “Arrendar”.

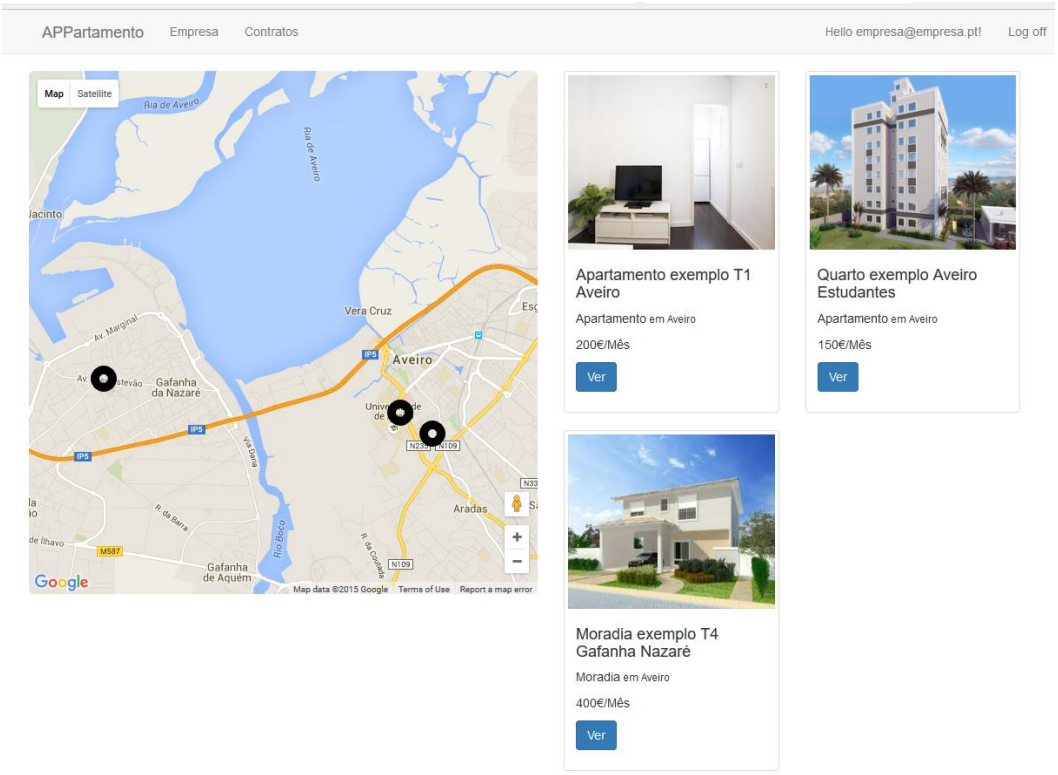


Figura 22 - Resultado pesquisa imóveis

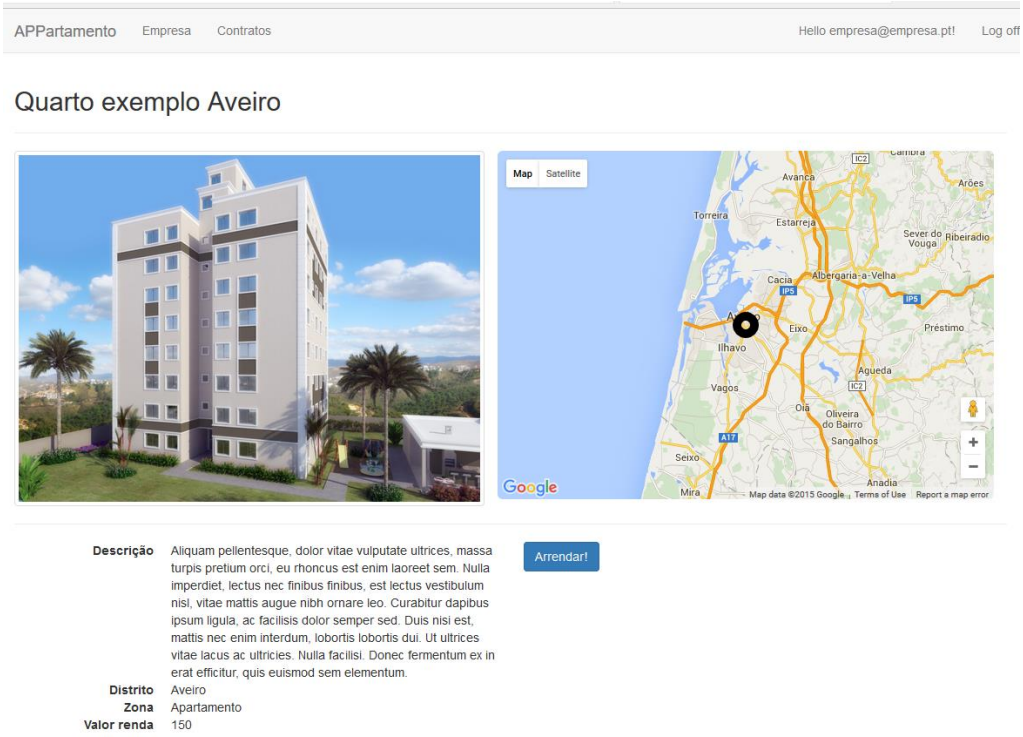


Figura 23 - Informações imóvel

A página de pedido (Figura 24) de contrato é a página final até o utilizador ter um pedido feito. Nesta página existem, mais uma vez, algumas informações básicas do imóvel e ainda os campos necessários para que seja possível pedir um contrato. Estes campos são a data de início e a data de fim, que representam quando o inquilino pretende começar e finalizar o contrato. Além disso, são ainda pedidos os documentos necessários. Neste momento ainda só é pedido o documento de identificação mas a aplicação está feita de modo a serem facilmente inseridos novos documentos requeridos para efetuar este passo. Por fim, o utilizador só tem de carregar no botão arrendar e o seu pedido de contrato estará feito.

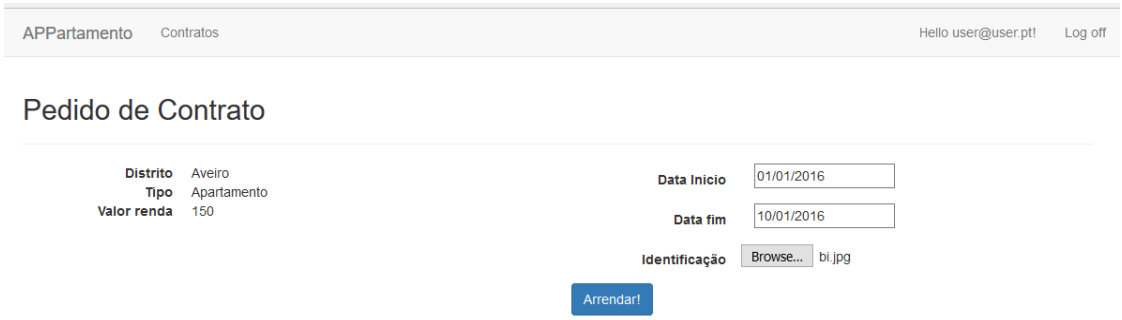


Figura 24 - Pedir contrato

4.3 GESTÃO CONTRATOS

Com o pedido de contrato feito, a entidade de arrendamento e o futuro inquilino podem ver na página contratos os seus contratos e o estado dos mesmos (Figura 25).

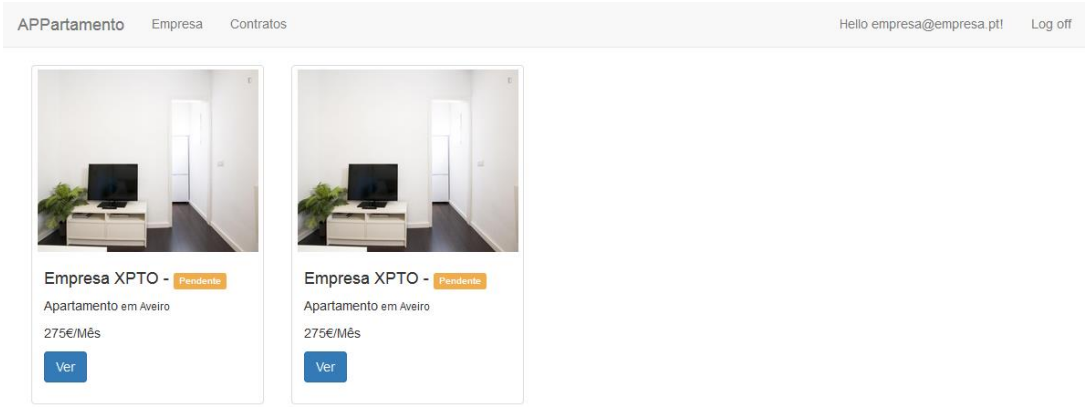


Figura 25 - Página contratos

Quando o gestor do imóvel clicar em "Ver" num dos contratos, ele chegará a uma nova página onde poderá decidir o próximo estado do contrato. Ainda dispõem de um sistema de mensagens para que ele e o inquilino possam comunicar (Figura 26). No Anexo 0 está disponível o fluxo completo dos estados implementados.

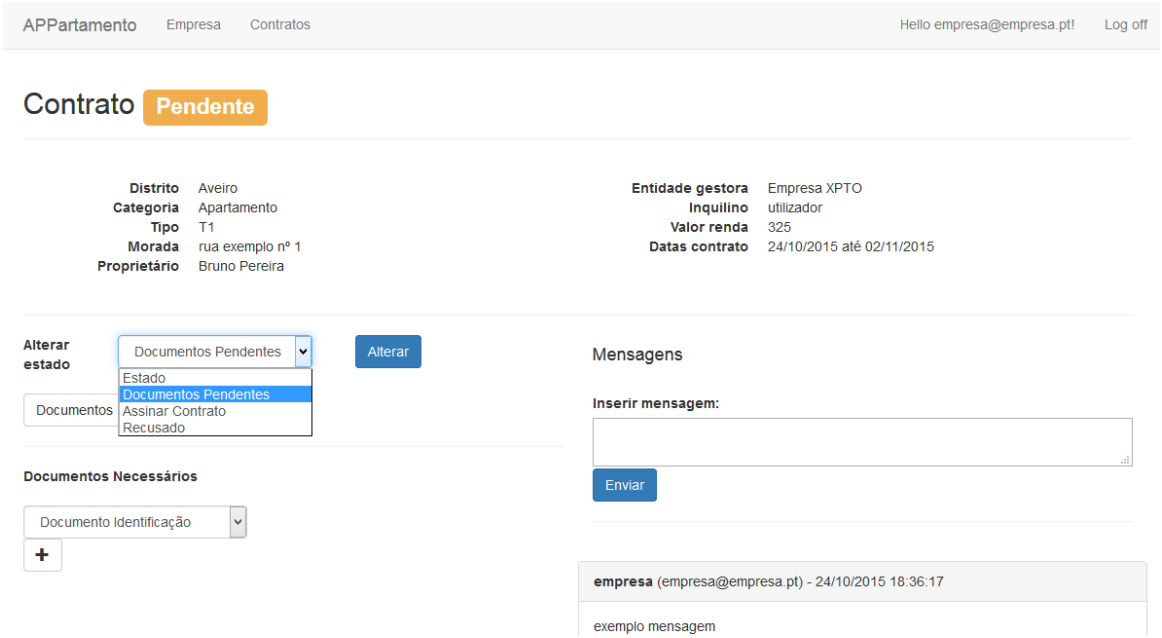


Figura 26 - Página contrato gestor imóvel

Depois da entidade de arrendamento estar satisfeita com todos os documentos que recebeu, este pode enviar um contrato ao futuro inquilino e pedir que este o assine.

O futuro inquilino poderá aceder então à página do contrato e estará disponível o contrato. Este, deverá descarregar o mesmo, assinar e enviar de volta para o sistema (Figura 27).

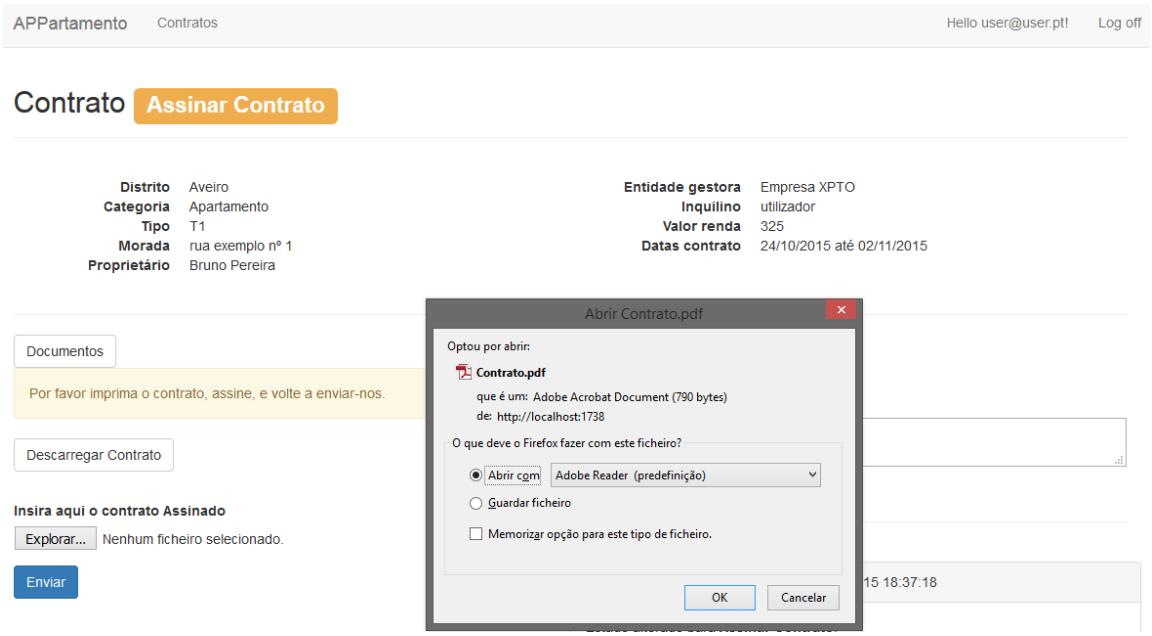


Figura 27 - Utilizador assinar contrato

Com o contrato assinado, a entidade de arrendamento poderá decidir entre aceitar, recursar ou até pedir novamente documentos. Caso o contrato seja aceite, será necessário que o inquilino faça o primeiro pagamento para que o contrato passe de aceite para ativo.

Na imagem abaixo podemos ver alguns exemplos de representação dos diversos estados de um contrato na página geral dos contratos.

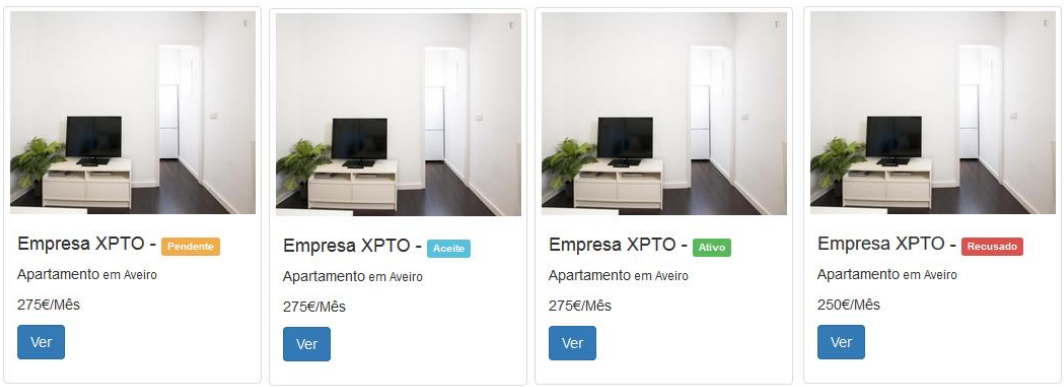


Figura 28 - Exemplos estados contratos

4.4 PAGAMENTOS

4.4.1 Pagamentos rendas

Com o contrato aceite, o inquilino irá ter acesso na página de informações do contrato, a uma tabela que lhe permite gerir os seus pagamentos. Esta tabela aparece ao clicar no novo botão "Pagamentos" que se encontra ao lado do botão que permite ver os documentos (Figura 29).

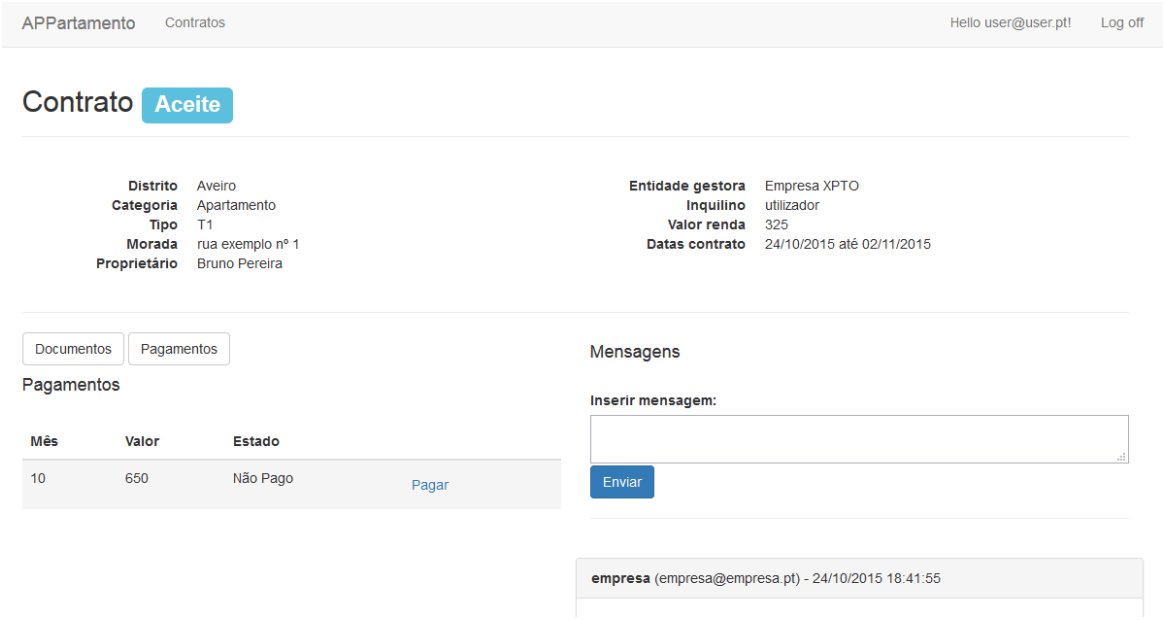


Figura 29 - Inquilino pagamentos

Esta tabela permite ao inquilino ver as rendas pagas e por pagar, o valor e o mês referente. Caso o pagamento não esteja feito, é disponibilizado um botão para que esse possa ser realizado. Esse botão redireciona o utilizador para uma página onde o pagamento pode ser realizado (Figura 30).

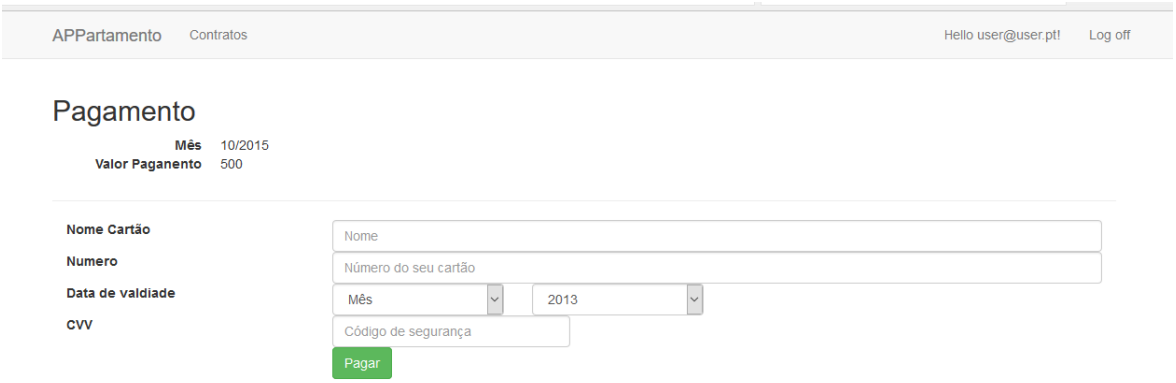


Figura 30 - Pagamento inquilinos

Com o pagamento feito, deixa de existir a opção "Pagar" e o inquilino passa a ter disponível o "Recibo" referente a esse pagamento. Ao clicar em "Recibo", o inquilino será redirecionado para uma página onde estará disponível para descarregar o mesmo. No caso do Moloni, esta nova página já não faz parte do sistema. É pedido à API o recibo e é esta que disponibiliza o endereço onde o utilizador pode descarregar o mesmo.

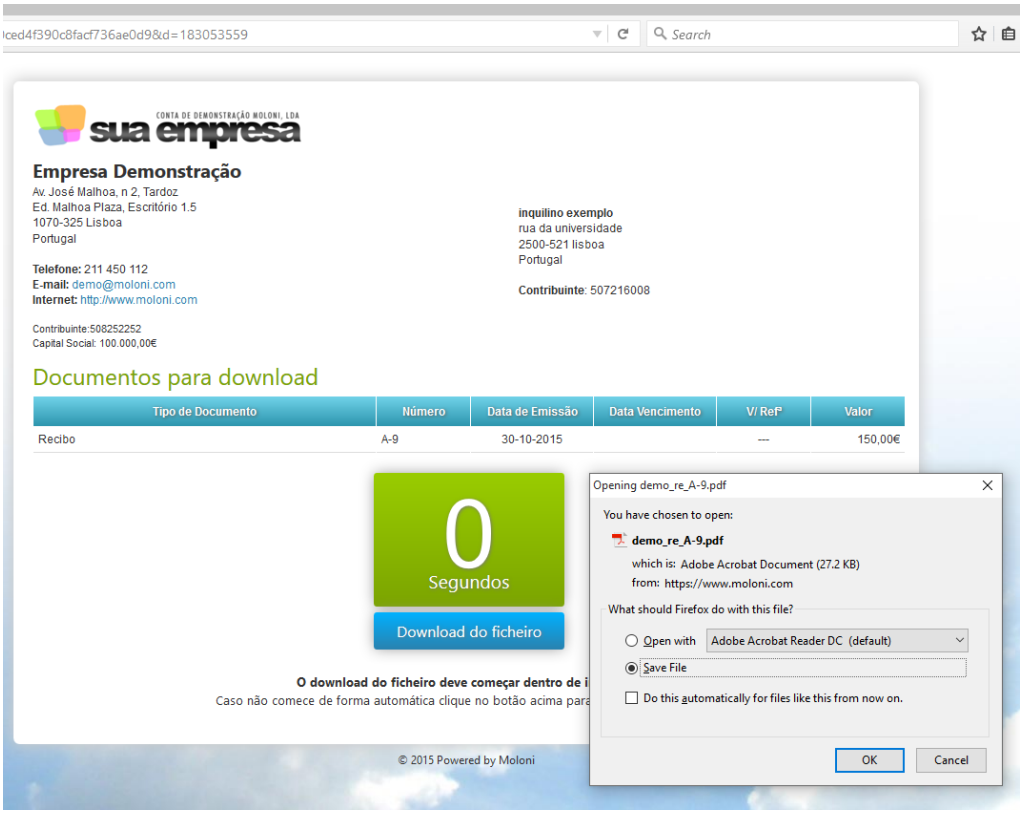


Figura 31 - Redireccionamento para descarregar fatura

4.4.2 Pagamentos do serviço

Por fim, é ainda importante referir a página de pagamentos das entidades de arrendamento. Este sistema fornece um serviço e, por este motivo, podem ser criadas várias formas e maneiras de cobrar aos utilizadores pelo serviço que lhes é fornecido. Assim, foram criados dois tipos de pagamento que a entidade de arrendamento pode escolher. "Renda" é um modo de pagamento que calcula uma pequena percentagem do valor das rendas dos imóveis com contrato ativo no atual mês. "Fixo" é outra possibilidade que cobra às entidades de arrendamento um valor fixo por cada imóvel que é inserido no sistema. Na Figura 32 podemos ver a página de definições onde as entidades de arrendamento podem fazer essa escolha.

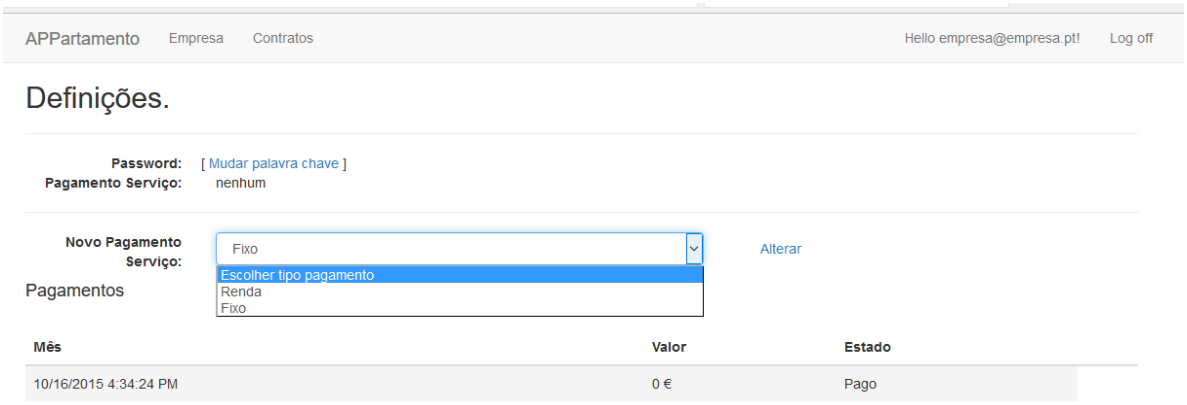


Figura 32 - Pagamentos gestores imóveis

5. CONCLUSÕES

O objetivo desta dissertação é o desenvolvimento de uma solução que permita a um conjunto de entidades fazer a gestão de arrendamento de imóveis. Esta solução tinha como requisitos permitir a um conjunto de entidades de arrendamento partilhar o mesmo sistema na realização de diversas tarefas como, gerir imóveis, gerir contratos e fazer a sua faturação no próprio sistema. Para isso, foi criado um sistema que será fornecido como serviço aos utilizadores. Este sistema permite às entidades de arredamento adicionarem os seus imóveis ao sistema e fazerem a publicitação dos mesmos. Utilizadores interessados em arrendar um imóvel publicitado podem fazer um pedido de contrato e caso seja aceite podem gerir os seus pagamentos e descarregar os recibos dos mesmos. No lado da entidade de arrendamento, é possível gerir diversos contratos simultaneamente e também gerir os pagamentos em relação ao consumo do serviço. A faturação é suportada pelo serviço fornecido pelo Moloni que permite a emissão de recibos através do consumo de uma API. A aplicação desenvolvida realiza os requisitos básicos necessários para a aplicação proposta nesta dissertação mas ainda existe muito trabalho a realizar para que esta aplicação possa ser usada comercialmente.

5.1 PROBLEMAS ENCONTRADOS

Durante o desenvolvimento desta dissertação foram encontrados alguns problemas, nos quais um dos principais foi o como estruturar os dados de maneira a que o sistema fosse capaz de suportar diversos serviços de faturação. Cada serviço de faturação tem os seus requisitos e estrutura de dados diferentes o que implica que a estrutura de dados seja capaz de lidar com essas diferenças sem afetar dados já existentes. Para resolver este problema optou-se por separar os campos que são comuns neste tipo de APIs dos que são mais específicos. Um exemplo de campo específico é, no caso do Moloni, o campo que

define o *template* dos documentos. Deste modo, quando é adicionado um novo serviço de faturação ao nosso sistema, caso seja necessário adicionar novos campos, as tabelas onde já existem dados de outros serviços de faturação não irão necessitar de ser alteradas.

Outro problema que foi encontrado ao longo do desenvolvimento foi como fazer o sistema criar automaticamente novos pagamentos na base de dados. Todos os meses é necessário fazer um novo pedido de pagamento aos inquilinos e, para tornar esse pedido automático, foi criado um serviço que, quando é chamado, verifica se existem novos pagamentos a serem criados. Para chamar o serviço está a ser usada uma consola do Windows. Esta consola corre uma *thread* que, consoante o intervalo de tempo definido, chama repetitivamente o serviço de verificação de novos pagamentos da nossa aplicação.

5.2 TRABALHO FUTURO

Apesar dos objetivos principais terem sido cumpridos, existe ainda trabalho a ser realizado. Em primeiro lugar, uma das áreas da aplicação que precisa de ser mais trabalhada é a dos pagamentos. Apesar de a aplicação ter o ato de pagamento e estes serem registados na base de dados, o pagamento propriamente dito, ou seja o suporte á transferência de dinheiro entre duas entidades não está implementado. Será necessário procurar e implementar um serviço de pagamentos *online* com o nosso sistema, como por exemplo o PayPal.

Outra área que precisa de ser trabalhada é o conceito de entidade de arrendamento e os membros que lhe pertencem. Neste momento a estrutura de dados suporta este conceito mas, a nível de lógica, ainda não está completamente implementado. Falta criar uma funcionalidade que permita aos administradores das entidades de arrendamento convidar ou aceitar membros, para que esses possam ajudar na gestão dos imóveis dessa entidade.

Em relação à base de dados, seria também interessante tentar encontrar uma forma de melhorar a estrutura das tabelas que suportam as informações das APIs de faturação. Apesar de ter sido encontrada uma solução para este problema, seria interessante voltar a analisar esta situação e tentar melhorar a estrutura das tabelas.

Neste momento apenas foi integrada uma API de faturação com o sistema, e como este sistema foi desenvolvido para suportar várias, era interessente começar a expandir o leque de APIs que os utilizadores têm disponíveis.

Seria também interessante a implementação de testes de *software* às diferentes funcionalidades do sistema a fim de garantir robustez do mesmo.

6. BIBLIOGRAFIA

[1] IMOVIRTUAL, “Procura imobiliária, por tipo de produto, em Portugal - Agosto,” 2015. [Online]. Available: http://www.imovirtual.com/estatisticas/procura_imobiliaria/. [Acedido em Setembro 2015].

[2] IMOVIRTUAL, “Estrutura da procura imobiliária, por finalidade - Agosto,” 2015. [Online]. Available: http://www.imovirtual.com/estatisticas/procura_imobiliaria/finalidade/. [Acedido em Setembro 2015].

[3] Uniplaces, “What is Uniplaces?,” 28 Julho 2015. [Online]. Available: http://help.uniplaces.com/hc/en-us/articles/201920358-What-is-Uniplaces-. [Acedido em Setembro 2015].

[4] Uniplaces, “Acomodações Lisboa,” Setembro 2015. [Online]. Available: https://www.uniplaces.com/accommodation/lisbon. [Acedido em Setembro 2015].

[5] Uniplaces, “What is the service fee?,” 28 Julho 2015. [Online]. Available: http://help.uniplaces.com/hc/en-us/articles/201920558-What-is-the-service-fee-. [Acedido em Setembro 2015].

[6] Imovirtual, “Anúncios de apartamentos para arrendar em Aveiro,” 2015. [Online]. Available: http://www.imovirtual.com/imoveis/apartamentos/arrendar/-/Aveiro/Aveiro/. [Acedido em Setembro 2015].

[7] IMOVIRTUAL, “Regras,” [Online]. Available: http://www.imovirtual.com/regras/. [Acedido em Setembro 2015].

[8] Moloni, “Sobre o Moloni,” 2015. [Online]. Available: https://www.moloni.com/about/index.php. [Acedido em Abril 2015].

[9] Moloni, “Developers,” 2015. [Online]. Available: https://www.moloni.com/dev/. [Acedido em Abril 2015].

[10] Moloni, “Painel Principal,” 2015. [Online]. Available: https://www.moloni.com/demo/Root/showAll/. [Acedido em Setembro 2015].

[11] InvoiceXpress, “O que é o InvoiceXpress?,” 2014. [Online]. Available: https://invoicexpress.com/faqs/ix/programa-facturacao-online-certificado. [Acedido em Abril 2015].

[12] InvoiceXpress, “Características,” 2014. [Online]. Available: https://invoicexpress.com/caracteristicas. [Acedido em Abril 2015].

[13] weoInvoice, “weoInvoice,” 2015. [Online]. Available: https://www.weoinvoice.com/index.php?module=home&func=plans. [Acedido em Abril 2015].

[14] InvoiceXpress, “Planos à medida do seu negócio,” 2014. [Online]. Available: https://invoicexpress.com/planos-precos. [Acedido em Setembro 2015].

[15] Moloni, “Moloni - Escolha o seu plano,” 2015. [Online]. Available: https://www.moloni.com/plans/index.php. [Acedido em Setembro 2015].

[16] Comité Técnico Fiscal da Moneris, “moneris,” 16 Abril 2015. [Online]. Available: http://www.moneris.pt/noticia.php?cod=1323. [Acedido em Setembro 2015].

[17] Diário da República, “Diário da República, 1.ª série N.º 252,” 31 Dezembro 2014. [Online]. Available: https://dre.pt/application/file/66015866. [Acedido em Outubro 2015].

[18] Diário da República, “Diário da República, 1.ª série N.º 252,” 31 Dezembro 2014. [Online]. Available: https://dre.pt/application/file/66014834. [Acedido em Outubro 2015].

[19] Autoridade tributária e aduaneira, “MANUAL DE INTEGRAÇÃO DE SOFTWARE,” 7 Julho 2015. [Online]. Available: http://info.portaldasfinancas.gov.pt/NR/rdonlyres/8C8E5D51-4F7D-4C09-86E2-070072D04227/0/Comunicacao_contratos_arrendamento_emissao_recibo_renda.pdf. [Acedido em Setembro 2015].

[20] M. Rouse, “what is multi-tenancy?,” Agosto 2014. [Online]. Available: http://whatis.techtarget.com/definition/multi-tenancy. [Acedido em Setembro 2015].

[21] R. Williams, “Web Forms, MVC, and Web Pages! Oh my!,” 8 Outubro 2013. [Online]. Available: http://www.codeproject.com/Articles/665118/Web-Forms-MVC-and-Web-Pages-Oh-my. [Acedido em Setembro 2015].

[22] Microsoft, “ASP.NET Overview,” [Online]. Available: https://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx#wf_mvc_wp. [Acedido em Setembro 2015].

[23] J. Chadwick, T. Snyder e H. Panda, Programming ASP.NET MVC 4, O’Reilly, 2012.

[24] w3schools, “ASP.NET Razor - Markup,” [Online]. Available: http://www.w3schools.com/aspnet/razor_intro.asp. [Acedido em Setembro 2015].

[25] R. Shannon, “Ross,” 21 08 2012. [Online]. Available: http://www.yourhtmlsource.com/starthere/whatishtml.html. [Acedido em Setembro 2015].

[26] “Lesson 1: What is CSS?,” [Online]. Available: http://html.net/tutorials/css/lesson1.php. [Acedido em Setembro 2015].

[27] S. Chapman, “What Is JavaScript?,” [Online]. Available: http://javascript.about.com/od/reference/p/javascript.htm. [Acedido em Setembro 2015].

[28] jQuery, “What is jQuery?,” 2015. [Online]. Available: http://jquery.com/. [Acedido em Setembro 2015].

[29] Bootstrap, “Bootstrap,” 2015. [Online]. Available: http://getbootstrap.com/. [Acedido em Setembro 2015].

[30] Google, “Google Maps Javascript API,” 13 Outubro 2015. [Online]. Available: https://developers.google.com/maps/documentation/javascript/tutorial. [Acedido em Julho 2015].

7. ANEXOS

7.1 TECNOLOGIAS

Neste capítulo irão ser analisadas as principais ferramentas que serão utilizadas no desenvolvimento da aplicação.

ASP.NET

ASP.NET é um *framework* para o desenvolvimento de aplicações web do lado do servidor. Existem várias tecnologias de desenvolvimento com ASP.NET: Web Forms, MVC e Web Pages. Web Pages é melhor para programadores sem tanta experiência pois tem uma curva de aprendizagem mais pequena, apesar disso podem ser feitas aplicações simples e complexas com a mesma [21]. Web Forms é um pouco mais complexa mas ainda assim é uma tecnologia que permite desenvolver aplicações rapidamente e é boa para aplicações de qualquer tamanho [21]. MVC é a tecnologia mais recente das três. É a mais difícil de aprender mas em troca oferece uma melhor flexibilidade e controlo das aplicações [21]. Um dos motivos pelo qual esta tecnologia facilita a gestão de projetos mais complexos é o facto de esta dividir as aplicações em três partes: *Model*, *Views* e *Controller*. Os três tipos de tecnologias de desenvolvimento referidos anteriormente, apesar de terem as suas diferenças, não são completamente independentes pois estes são baseadas na mesma *framework*, assim por exemplo em MVC poderão ser utilizadas ferramentas de Web Pages [21] [22] [23].

Razor

Razor é uma linguagem de marcação que permite adicionar código do lado do servidor numa página web. Quando uma página é pedida, o código Razor é executado no servidor antes de a página ser enviada para o cliente. Com isto é possível fazer algumas tarefas mais complexas e tornar as páginas mais dinâmicas [24].

HTML

HTML ou *HyperText Markup Language* é uma linguagem de marcação que é utilizada principalmente para o desenvolvimento de páginas *Web*. É uma linguagem fácil de usar que permite a estruturação de conteúdo com um paradigma de blocos. Essa estruturação é construída através da inserção de pequenos códigos ou *tags* num ficheiro com conteúdo texto. Quando esse ficheiro é gravado no formato HTML e visualizado através de um *browser*, o texto aparece formatado com a estrutura e características das *tags* inseridas [25]. Diferentes *tags* produzem diferentes efeitos na altura de formatar o conteúdo pretendido e, no resultado da formatação, essas *tags* não aparecem. Apesar de ser uma das principais linguagens no desenvolvimento de páginas *Web*, como se foca na estruturação do conteúdo, só por si apenas permite a criação de páginas visualmente rudimentares. Por este motivo esta é uma linguagem que normalmente é complementada com outras linguagens e tecnologias. Podemos ver em baixo um exemplo de uma página HTML básica:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Exemplo</title>
    </head>
    <body>
        Exemplo HTML
    </body>
</html>
```

CSS

CSS ou *Cascading Style Sheets* é uma linguagem que permite a definição do estilo de documentos que usam linguagens de marcação como o HTML. O CSS permite definir tipos

de letra, cores, tamanhos, posições, margens, entre outras coisas [26]. Esta linguagem surge como uma ferramenta para aumentar a capacidade de customização e de controlo sobre as características de aspeto dos elementos de um documento. Além disso, esta linguagem permite que as características de um documento sejam partilhadas com outros documentos. Em baixo temos um exemplo de customização de um elemento HTML:

```
body {
    padding-top: 50px;
    padding-bottom: 20px;
}
```

Javascript

Javascript é uma linguagem de programação que corre do lado do cliente com a principal função de tornar as páginas web mais interativas e dinâmicas [27]. Javascript é uma das linguagens mais populares do mundo, está normalmente associada a páginas HTML, para realizar funções como paginação, gerir eventos como o *click* do rato e controlar os comportamentos de formulário. Hoje em dia esta linguagem é suportada por todos os principais *Web browsers* e é uma linguagem essencial para o enriquecimento de páginas web.

Jquery

Jquery é uma biblioteca que adiciona um conjunto de características ao Javascript. Esta biblioteca tem como objetivo ajudar a tornar a utilização de Javascript em páginas HTML mais fácil e rápido [28].

Bootstrap

Bootstrap é um *front-end framework* que disponibiliza um conjunto de ferramentas para facilitar o desenvolvimento de interfaces em páginas e aplicações *Web*. O Bootstrap permite ajudar a estruturar as páginas web e ainda oferece um conjunto de estilos para os diversos elementos HTML, o que facilita o desenvolvimento do aspeto da interface [29].

Google Maps APIs

Gooogle Maps APIs é uma ferramenta que permite integrar mapas na nossa aplicação. É uma ferramenta fácil e rápida de começar a usar, que pode ser integrada numa página *Web* através da API do Google usando Javascript [30].

MSSQL

MSSQL abreviatura para Microsoft SQL Server é um sistema de gestão de bases de dados relacionais. Esta ferramenta tem como principal funcionalidade guardar e disponibilizar a informação consoante as necessidades de uma aplicação. Abaixo um exemplo de uma *query* a base de dados:

```
SELECT * FROM Contrato
```

7.2 MODELO DADOS COMPLETO

Na Figura 33 podemos ver o diagrama completo da base de dados. Neste anexo irão ser ainda analisadas mais algumas tabelas da base de dados.

Tabela EntidadeGestores

A tabela **EntidadeGestores** é responsável por fazer a relação entre os imóveis e as entidades. Esta surge com a necessidade de permitir que as empresas tenham utilizadores responsáveis pelos seus imóveis.

Atributo	Descrição
EntidadeGestoresId	Identificador único de cada entidade gestora da tabela
Nome	Identificador único do imóvel do contrato

Tabela 4 - Atributos tabela EntidadeGestores

Tabela DefinicoesEntidade

A tabela **DefinicoesEntidade** é responsável por guardar as definições feitas por uma entidade caso esta as tenha.

Atributo	Descrição
EntidadeId	Identificador único de cada entidade
FaturaçãoPropria	Se esta entidade utiliza a faturação do nosso sistema
softwareGestaoId	Identificador único do serviço de faturação usado (caso exista)
SGUserId	Identificador único de utilizador de *software* de faturação
DefinicoesId	Identificador único da linha da tabela
ModalidadeId	Qual a modalidade de pagamento escolhida pela entidade

Tabela 5 - Atributos tabela DefinicoesEntidade

Tabela SoftwareGestaoUser

A tabela **SoftwareGestaoUser** é responsável por guardar os parâmetros genéricos considerados em todas as APIs de faturação. Esta tabela é ligada com a tabela **SoftwareGestaoUserTokens** e com a tabela **SGMoloniUserSettings**. A tabela **SoftwareGestaoUserTokens** é responsável por guardar os *tokens* que são necessários para a autenticação nas APIs e a **SGMoloniUserSettings** é uma tabela específica para guardar dados apenas da API do Moloni.

Atributo	Descrição
email	Correio eletrónico do utilizador
password	Palavra-chave do utilizador da API
language_id	Identificador único da linguagem na API
company	Nome da empresa
vat	Número de identificação fiscal
slug	Nome utilizador na API
country_id	Identificador único do país na API
user_id	Identificador único do utilizador na API
company_id	Identificador único da empresa na API
SGUserId	Identificador único de utilizador de *software* de faturação

Tabela 6 - Atributos tabela SoftwareGestaoUser

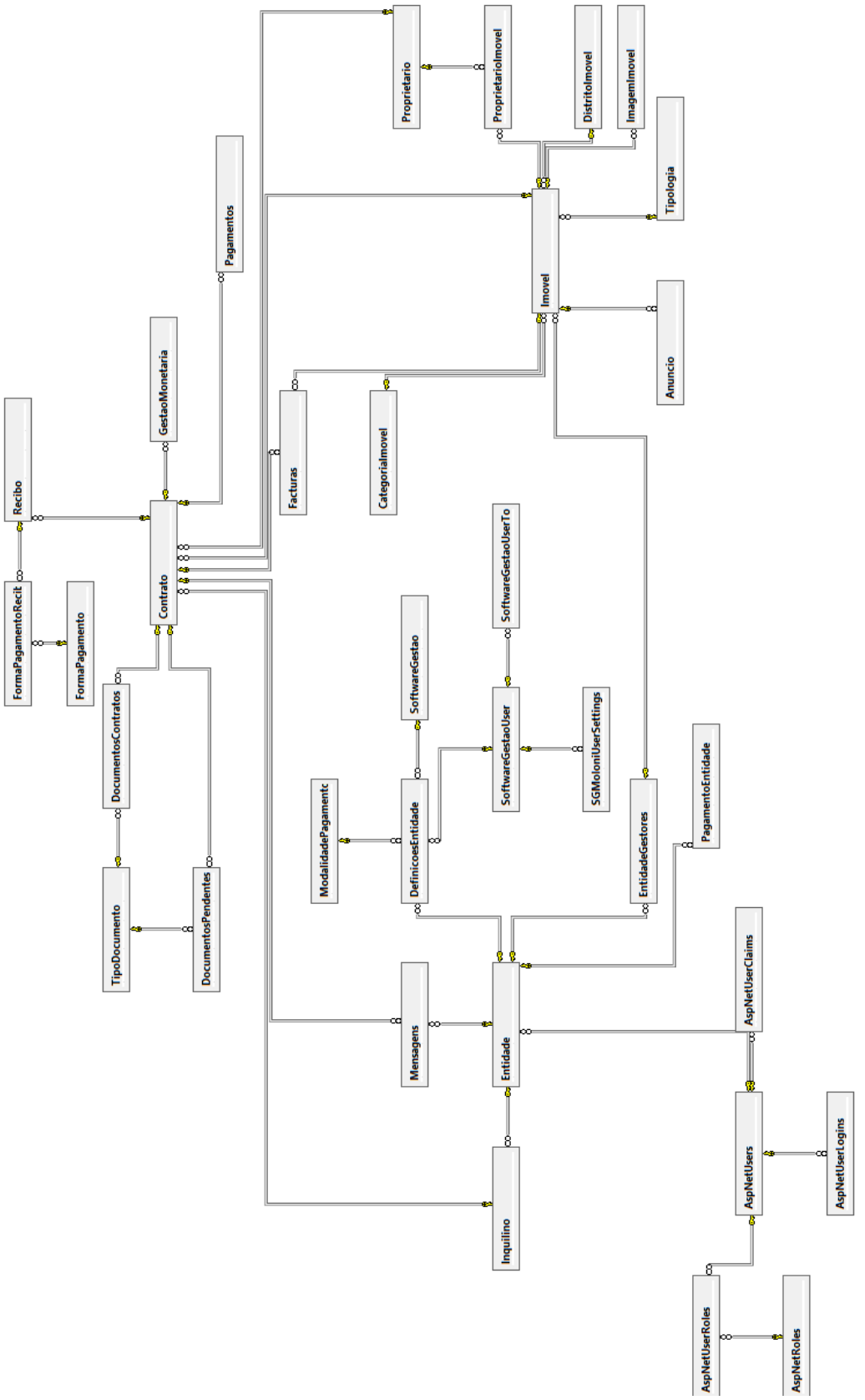


Figura 33 - Modelo dados

7.3 ESTADOS CONTRATO

Neste anexo pode ser visto o fluxo de um contrato na nossa aplicação.

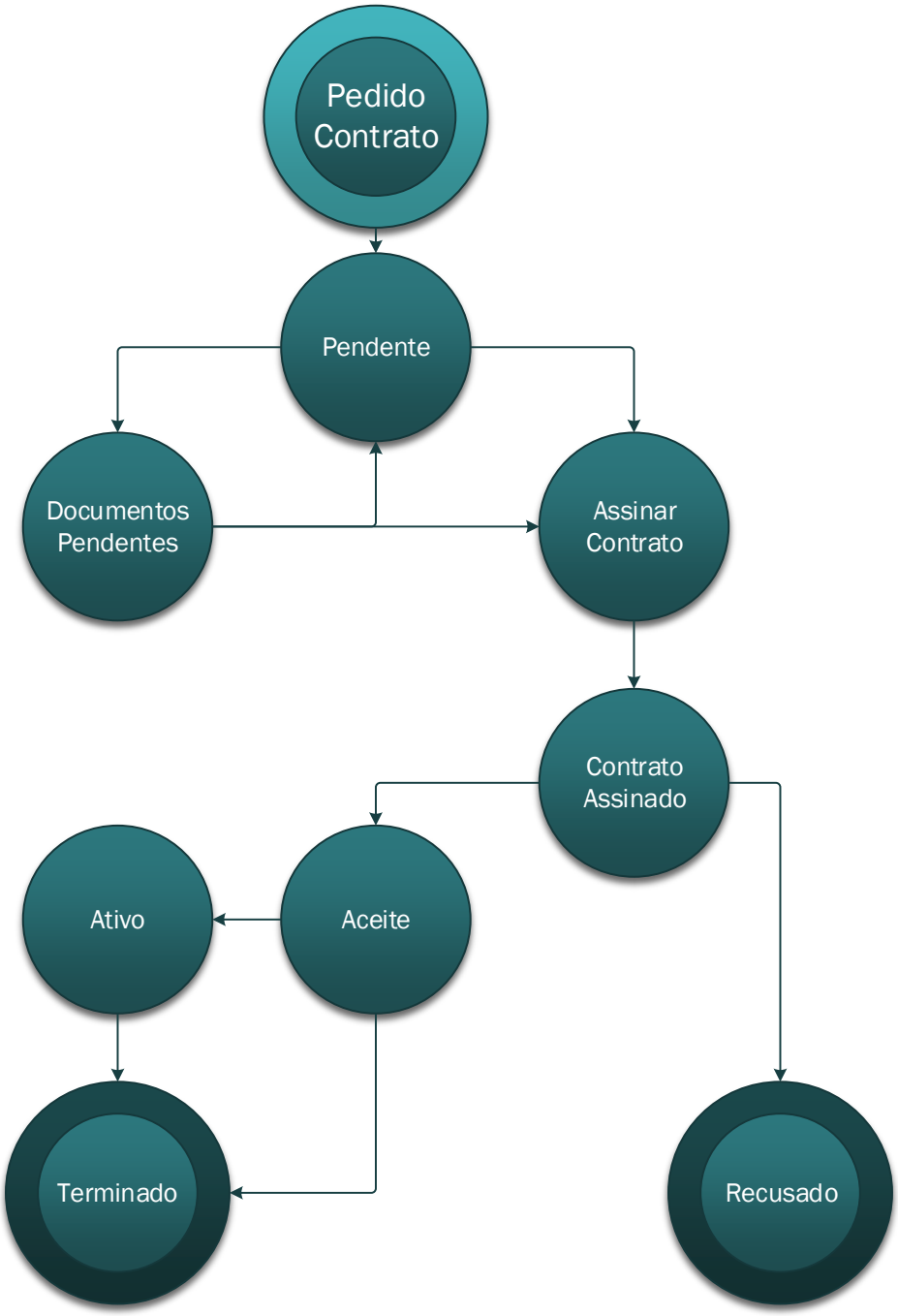


Figura 34 - Fluxo estados contrato

7.4 PEDIDOS MOLONI

Na Figura 35 podemos ver um exemplo de um pedido à API do Moloni e a resposta que foi recebida. Neste caso o pedido feito foi de obtenção do documento de um recibo, e na resposta podemos ver o endereço onde este se encontra disponível.

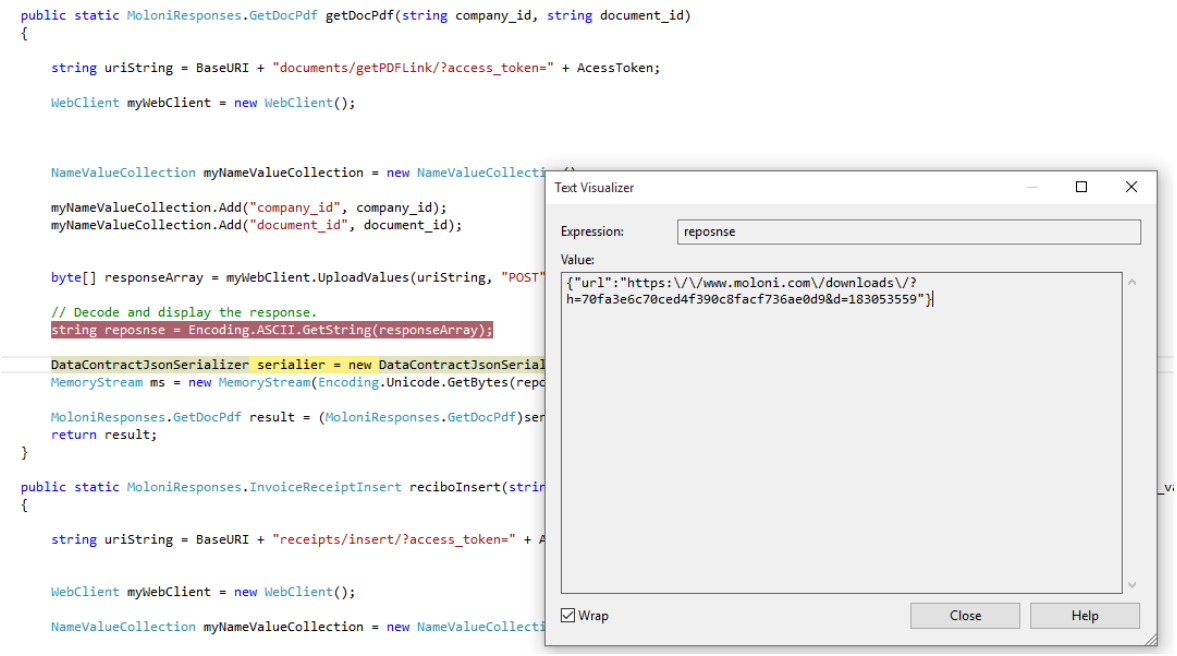


Figura 35 - Resposta Moloni com URL de recibo

No código abaixo podemos ver um exemplo de uma função da classe que faz os pedidos à API do Moloni.

```
        public static MoloniResponses.CompanyGetAll CgetAll()
        {
            String url = BaseURI + "companies/getAll/?access_token=" + AcessToken;

            WebClient myWebClient = new WebClient();

            NameValueCollection collection = new NameValueCollection();
            byte[] responseArray = myWebClient.UploadValues(url, "POST", collection);
            string response = Encoding.ASCII.GetString(responseArray);

            DataContractJsonSerializer serialier = new
DataContractJsonSerializer(typeof(MoloniResponses.CompanyGetAll));
            MemoryStream ms = new MemoryStream(Encoding.Unicode.GetBytes(response));

            MoloniResponses.CompanyGetAll result =
(MoloniResponses.CompanyGetAll)serialier.ReadObject(ms);

           return result;
        }
```

Para realizar pedidos ao Moloni, é usada a classe **WebClient** do .NET que permite fazer pedidos HTTP, neste caso específico, pedidos POST. No caso do código acima é pedido ao Moloni para enviar todas as empresas que já foram inseridas. Para fazer este pedido é necessário o construir o URL para onde vai ser enviado o pedido. Depois do pedido feito o Moloni envia uma resposta em formato JSON. Para que a informação fique numa estrutura que o nosso sistema consiga interpretar, ela é serializada usando a classe **DataContractJsonSerializer** do .NET, ficando assim a informação pronta a ser consumida. Em baixo fica o exemplo da classe que define a estrutura da resposta obtida pelo Moloni para o pedido feito a cima.

```
[DataContract]
public class CompanyGetAll
{
    [DataMember]
    public string email { get; set; }
    [DataMember]
    public int company_id { get; set; }
    [DataMember]
    public string name { get; set; }
    [DataMember]
    public string vat { get; set; }
    [DataMember]
    public string address { get; set; }
    [DataMember]
    public string city { get; set; }
    [DataMember]
    public string zip_code { get; set; }
    [DataMember]
    public int country_id { get; set; }
    [DataMember]
    public string image { get; set; }
    [DataMember]
    public Country country { get; set; }
}
```

7.5 CRIAÇÃO DOS PDF'S

Para a criação de documentos em formato PDF foi utilizada uma biblioteca externa chamada iTextSharp. Esta biblioteca permite criar documentos PDF de raiz e foi utilizada para criação dos documentos de contratos necessários para o nosso protótipo. Em baixo pode ser visto um exemplo da criação básica de um documento PDF usando esta biblioteca.

```
Document pdfDoc = new Document();

PdfWriter writer = PdfWriter.GetInstance(pdfDoc, new FileStream(…));
pdfDoc.Open();

pdfDoc.Add(new Paragraph("Texto exemplo pdf"));
PdfContentByte cb = writer.DirectContent;
cb.Stroke();

pdfDoc.Close();
```